Optimised Ray Tracing for the SuperNEC Implementation of the Uniform Theory of Diffraction

Robert Hartleb

A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirement for the degree of Master of Science in Engineering.

Johannesburg, July 2006

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Science in Engineering in the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

_____

Robert Hartleb

This _____ day _____ of 2006

Abstract

Geometric optimisations are presented for the UTD in SuperNEC which is a commercial electromagnetic software package. Path finding optimisations rapidly find propagation paths of electromagnetic waves by using back face culling to determine the visible plates of polyhedral structures and by using reflection and diffraction zones which use image theory and the law of diffraction to determine illuminated spatial regions. An octree reduces the number of intersections during the shadow tests. Numerical results show that overall the optimisations halve the run time of the software for models which consist of plates and cylinders. The path finding optimisations do not scale with model size, are limited to plates and introduce errors. The mean absolute error due to the path finding optimisations is on average 0.02 dB for first order rays and 0.17 dB for second order rays. The octree optimisation scales with model size, can be used with any geometry and any type of ray and does not cause errors.

Acknowledgements

I would like to thank Professor Alan Clark for teaching me to first get an LED to flash and the demons and angels in the Wartenweiler Library for their indefatigable assistance.

Foreword

This dissertation is presented to the University of the Witwatersrand, Johannesburg for the degree of Master of Science in Engineering.

The dissertation is entitled "Optimised Ray Tracing for the SuperNEC Implementation of the Uniform Theory of Diffraction". In this study, research was undertaken to explore ray tracing methods to optimise the UTD implementation in SuperNEC. Various techniques where investigated to optimise the path finding and shadow test algorithms. Back face culling and visibility regions for reflections and diffractions where investigated for optimising path finding. Octrees are used for fast shadow tests. The optimisations where tested and implemented in C++ and tested in SuperNEC. Tests models are used to determine the performance of the optimisations and to compare the optimisations with the original algorithms in SuperNEC.

This document complies with the university's *paper model* format. The paper contains the main results of the research. The appendices present in detail the work conducted during the research.

Appendix A presents analysis of the problems with the current SuperNEC UTD implementation.

Appendix B presents the path finding optimisations. The principles behind the optimisations are explained and results of tests are presented.

Appendix C presents the shadow test optimisations. The octree used in the optimisations is presented together with empirical results from tests.

Appendix D presents results of the path finding and shadow optimisations used collectively on test models.

Appendix E lists the files stored on the attached compact disc. The files contain the source code of the optimisation techniques.

Contents

List of Figures

List of Tables

Part I

Paper: Optimised Ray Tracing for the SuperNEC Implementation of the Uniform Theory of Diffraction

Optimised Ray Tracing for the SuperNEC Implementation of the Uniform Theory of Diffraction

Robert Hartleb

***Abstract*— Geometric optimisations are presented for the UTD in SuperNEC which is a commercial electromagnetic software package. Path finding optimisations rapidly find propagation paths of electromagnetic waves by using back face culling to determine the visible plates of polyhedral structures and by using reflection and diffraction zones which use image theory and the law of diffraction to determine illuminated spatial regions. An octree reduces the number of intersections during the shadow tests. Numerical results show that overall the optimisations halve the run time of the software for models which consist of plates and cylinders. The path finding optimisations do not scale with model size, are limited to plates and introduce errors. The mean absolute error due to the path finding optimisations is on average 0.02 dB for first order rays and 0.17 dB for second order rays. The octree optimisation scales with model size, can be used with any geometry and any type of ray and does not cause errors.**

***Index Terms*— uniform theory of diffraction, geometrical optimisations, ray tracing.**

I. Introduction

THE uniform geometric theory of diffraction (UTD) [1], [2] is a theory of electromagnetic wave propagation based on ray methods [3]. Infinitesimally thin rays approximate the propagation paths of electromagnetic waves. When a ray intersects an object a reflection, diffraction or transmission occurs which continues propagating the wave. Geometric laws determine the paths of the rays and the UTD is applied to the paths to determine the electromagnetic field at any point on the ray. At an observer point the total field is the superposition of the fields of all the rays through that point. Using UTD the characteristics of antennas in the presence of complicated objects can be simulated.

SuperNEC [4], an antenna simulation program based on the method of moments (MoM) [5], includes a UTD implementation. An antenna is made up of MoM segments and objects in the presence of the antenna are represented by cylinders and plates. The interaction between the MoM segments is by means of rays. Far field radiation patterns are determined using the UTD and ray methods which propagate the electromagnetic fields from the segments to the far field by the successive application of the laws of reflection and diffraction.

Although the UTD is a simple and elegant propagation method, one of the biggest disadvantages is the computational expense due to the large numbers of rays required. In a model which consists of many segments and components, multiple ray paths exist from the segments to points in the far field. For each segment and far field point all geometrically valid rays must be found which means a large number of propagation paths must be examined. The UTD cannot be used with large models unless optimisation techniques are implemented to reduce the large number of rays.

Two parts of SuperNEC-UTD are slow: path finding and shadow tests [6]. In path finding the spatial coordinates of propagation paths are determined. Shadow tests determine if a propagation path is intersected by an object.

Ray tracing is a rendering technique used in computer graphics to generate realistic 2D images of 3D scenes [7]. Rays represent the propagation paths of light in the presence of 3D objects from a light source onto a 2D screen. A variety of techniques from ray-tracing have been used to optimise the UTD. Gutierrez *et al.* [8] describe the use of octree and binary space partitioning techniques to speed-up shadow tests. In the same article, a buffer is used to determine in advance the various components illuminated by rays. In [9] a visibility graph is constructed to determine the spatial regions which are illuminated by a transmitter either by direct rays or by combinations of reflections and diffractions. These data structures quickly identify geometrically valid paths. UTD has been used in conjunction with ray-tracing in various situations to determine the antenna radiation off aeroplanes [10], designing in indoor [11], [12] and outdoor [13] radio systems and to predict electromagnetic radiation in tunnels [14].

Four optimisations are presented that are divided into path finding and shadow test optimisations. The 3 path finding techniques reduce the time to find the coordinates of geometrically valid propagation paths. In [15] back face culling reduces the processing time of polyhedrons during ray tracing. Back face culling determines which plates of a polyhedron are visible from a segment. Reflection zones use image theory to determine the spatial regions which are illuminated by reflecting rays [16]. The diffraction zones use the law of diffraction [2] to define the spatial region which can be illuminated by a diffracting ray. Concepts similar to the reflection and diffraction zones are used by [16], [17], [18] to optimise indoor and outdoor propagation prediction software.

Octrees are used to optimise the shadow tests. Octrees were used by Glassner to optimise ray tracing [19] by exploiting the spatial coherence [20] of 3D models. An octree is a hierarchical decomposition of a spatial region. The region is recursively subdivided into a 3 dimensional grid of cubes of varying resolution. Given a point in space, the objects closest to the point are quickly retrieved. The subdivision of a region is stored in a kd-tree called an octree.

The causes of the high run times are presented in section II. The principles of the 4 optimisation techniques and results of tests are discussed in sections IV to V. In section VI the results of the combined performance of the optimisations are presented. Section VII objectively reviews the optimisations.

Fig. 1. Block diagram of original SuperNEC-UTD.

II. Problem Statement

Figure 1 shows the original structure of the SuperNEC-UTD methodology. The inputs are the coordinates of an MoM segment and an observer point. The electromagnetic field at the observer due to all rays through the observer point is the output. Path finding and shadow tests are explained in the next section. The final step in SuperNEC-UTD is calculating the electromagnetic fields along the valid ray paths.

A. Path Finding

Path finding searches for all geometrically valid propagation paths between 2 points [6]. Figure 2 shows an example

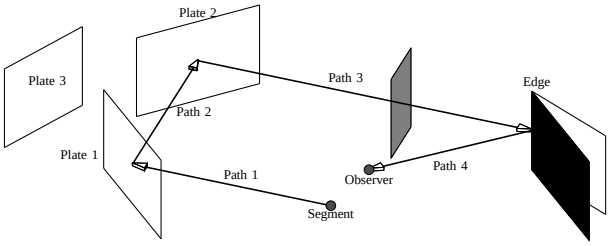


Fig. 2. Ray from a segment to an observer.

propagation path from a segment, reflecting in 2 plates and diffracting on an edge. Path finding determines the geometrical coordinates of rays by applying the laws of reflection and diffraction.

Path finding uses a brute force algorithm which does not scale with model size. Figure 2 shows one possible path. A path eg. from plate 2, to plate 1 and to plate 3 is also possible. There are many combinations of paths and the algorithm exhaustively examines all combinations (called the search space). As the model size increases the search space increases exponentially which leads to excessive run times. Often the ray paths are not valid because the laws of reflection and diffraction are violated so that path finding outputs only a few valid rays.

B. Shadow Tests

The shadow algorithm exhaustively searches all components to determine if a ray is blocked [6]. In figure 2, the ray is blocked by the shaded plate.

Every extra component added to a model increases the number of ray paths examined during path finding and in general increases the number of geometrically valid rays that need to be examined during the shadow tests. Adding an extra component also increases the number of intersection tests for each ray. These 2 factors lead to an exponential rise in the run time of the shadow algorithm with model size.

C. Path finding and shadow tests

The relative importance of the run times of the 2 algorithms varies depending on the rays. This is discussed in detail in [6]. If only line of sight rays and rays which only reflect or diffract *once* before propagating to the observer are used, the shadow tests dominate the run time because the search space is small. In this case the un-scalability of the path finding algorithm is not problematic. Only the shadow tests need to be optimised.

If second and higher order rays are used which include any 2 or more arbitrary combinations of a reflection or a diffraction (eg. a ray diffracting around a cylinder and then reflecting in a plate before propagating to the observer) path finding time will be much higher than the shadow test time. The complexity of the rays means examining a large search space to find geometrically valid rays.

III. Overview of the Optimisation

From the previous section, the principle aim of the optimisation techniques must not only be the reduction in the run time but also the scalability with model size of both the path finding and shadow algorithms. Geometric optimisations are proposed, namely path finding optimisations [21] which reduce the search space primarily for second and higher order rays (although first order rays are also optimised) and octree optimisations [22] which make the run time of the intersection tests independent of model size. The optimised structure of

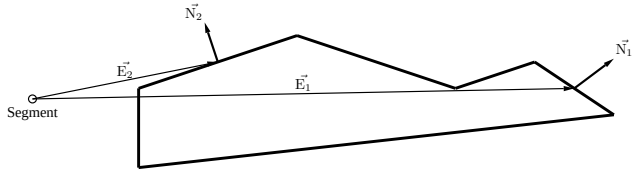


Fig. 3. Back face culling illustrated for the 2D case. Adapted from [15]

SuperNEC-UTD is shown in figure 4. All optimisations are time-space trade-offs. Each technique creates a data structure in a preprocessing stage. During path finding the data structures reduce the search space to geometrical regions where the laws of reflection and diffraction are not violated. The data structure for the shadow tests is used to retrieve those components which lie close to and which have a high probability of blocking a propagation path.

IV. Path finding optimisations

A. Back face culling

Back face culling [15] (BFC) eliminates those plates of a polyhedron which are not visible from a segment. Figure 3 illustrates the concept for the 2D case. The angle between the normal of a plate and the vector from the segment to a point on the plate determines if the plate is visible. If the plate is viewed from the front the angle between the 2 vectors is zero degrees. If the plate is viewed edge-on the angle is either -90° or 90°. Therefore a plate is visible if the angle is in the range -90° to 90°.

In figure 3 the plate with normal $\vec{\mathrm{N}}_1$ is not visible because the angle between $\vec{\mathrm{E}}_1$ is greater than 90°. The plate with normal $\vec{\mathrm{N}}_2$ is visible because the angle is smaller than 90°.

During path finding, rays are only traced to the plates of polyhedrons which are visible. BFC can only be used with polyhedrons.

BFC introduces errors when 2 connected plates of a polyhedron are not visible. In such a case a corner diffracted ray is possible at the corner of a common edge but because the plates are not visible in BFC, the ray is eliminated.

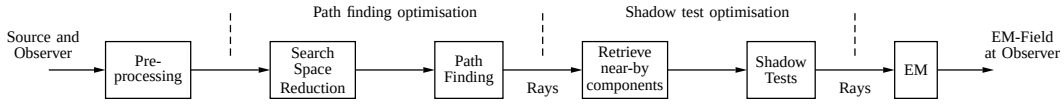


Fig. 4. Block diagram of optimised SuperNEC-UTD.

B. Reflection zones

Reflection zones [16] (RZ) use image theory to determine during the preprocessing stage the regions which are illuminated by a ray reflecting in a plate.

In figure 5, all reflections from a segment in a plate (P1) seem to originate from the image of the segment in the plate. The image lies on the line perpendicular to the plate and which connects the segment and image. The image is the same distance from the plate as the segment is from the plate. The spatial region called a reflection zone (shaded grey) is the semi-infinite area inside the imaginary lines extended from the image past the vertices of the plate.

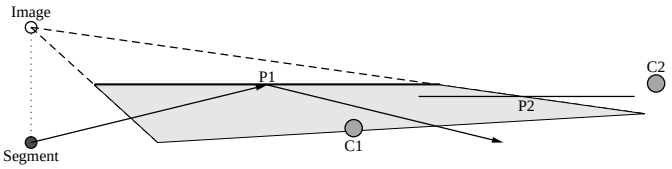


Fig. 5. Reflection zone for the 2D case. Adapted from [16].

Only reflections inside the zone are valid and it is possible to determine in advance if a point or a component can be reached by a reflection. Higher order reflection zones determine which regions are illuminated by multiple reflections. Reflection zones only work if the ray propagates directly from the segment to a plate. For higher order rays, the subsequent reflections must follow directly after the first reflection. If the ray diffracts or reflects in a cylinder before reflecting in the plate, the RZ cannot be used. During the pre-processing stage the image of a segment is used to construct the RZ. For diffractions and cylinder reflections the image cannot be calculated in advance.

C. Diffraction zones

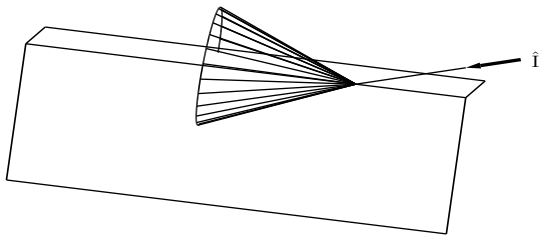


Fig. 6. 3D dimensional diffraction zone around a wedge. Adapted from [23].

Using the theory of diffraction as described by Keller [2], for each wedge (a common edge of 2 connected plates) a spatial region called a diffraction zone [16], [17] is determined. A diffraction zone (DZ) is the volume that is illuminated by a ray diffracting on an edge. Using a diffraction zone geometrically invalid rays are quickly eliminated.

In figure 6, a ray is incident on an edge which results in infinitely many diffracted rays which form a cone around the wedge [2]. Depending on the location of the incident ray, the diffracted ray is limited to 2 spatial regions: either above the wedge as shown in figure 6 or below the wedge. Path finding only propagates diffracted rays to points and components located in the diffraction zones.

A ray incident on the corner of a wedge can propagate underneath the wedge. Using DZ corner diffraction is not possible. Corner diffracted rays eliminated in this way introduce errors.

TABLE I
MODELS FOR PATH FINDING.

Model	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Plates (total)	7	14	21	28	35	42	49	56
Polyhedrons	1	1	1	1	2	2	3	3
Plates	6	6	20	6	12	32	38	32
Wedges	16	32	34	54	70	88	104	124
Edges	28	50	64	93	121	154	182	218
Single plates	1	2	1	3	4	4	5	4
Cylinders	1	1	2	2	3	3	4	4

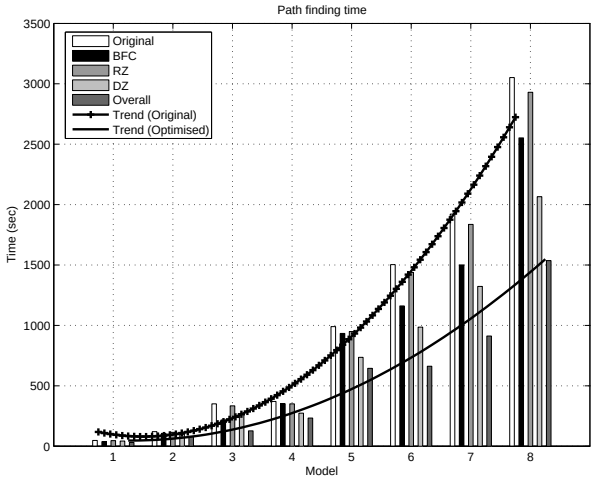


Fig. 7. Path finding time.

D. Results

Eight generic models listed in table I are used to test the path finding optimisations. For each model three 2D radiation patterns are calculated. Each path finding optimisation works only when specific geometries are present. BFC works only with polyhedrons. DZ work only with wedges (as found in polyhedrons or structures which consist of 2 or more connected plates). RZ work with all plates. The models are as generic as possible and consist of various geometrical structures. In table I, the total number of plates are listed. The number of polyhedrons are listed together with the number of plates used for those structures. Below the number of wedges are the total number of edges in the model. The number of single plates and the number of cylinders are listed at the bottom of the table.

1) Performance: The path finding time is shown in figure 7. Each group of 5 bars represents one model. Five bars show the results for the original program, the results for the 3 techniques

TABLE II
PATH FINDING SPEED-UP FACTORS ($\frac{\text{original time}}{\text{optimised time}}$)

Model	1	2	3	4	5	6	7	8
BFC	1.27	1.13	1.61	1.06	1.06	1.29	1.28	1.20
RZ	1.04	1.05	1.05	1.06	1.04	1.05	1.05	1.04
DZ	1.11	1.22	1.31	1.36	1.35	1.53	1.45	1.48
Together	**1.53**	**1.47**	**2.76**	**1.59**	**1.54**	**2.27**	**2.11**	**1.99**

run individually and the results using all 3 techniques together. Table II gives the individual and collective speed-up factors.

Examining the results shows that the original program has run time which increase exponentially with model size. The optimised program on average halves the run time but the increase remains exponential. In general DZ have the largest speed-up which increases with model size. As the model size increases there are more wedges which are used to build DZ. In the original program the increasing number of wedges causes an exponential growth in the number of diffraction rays. DZ limit the number of diffracted rays at each wedge thus limiting the exponential ray growth. This is shown by the reduced run times for DZ in figure 7 and by the speed-ups in table II. On average the DZ speed-up is about 1.35 which is a 26 % reduction in the run time.

BFC is less effective because the technique only identifies visible plates of polyhedrons. The speed-up results vary with the number of polyhedrons. Models with a large number of polyhedrons have a high BFC speed-up. The average speed-up using BFC is 1.24 which is a 19 % reduction in the run time.

RZ are the least effective in reducing the run time. The average speed-up is 1.05 which is a 5 % reduction in the run time. The speed-up is fairly constant for all models because the technique works with all plates.

On average the 3 techniques collectively reduce path finding time by a factor of 1.90 which is 47 % less time than the original program.

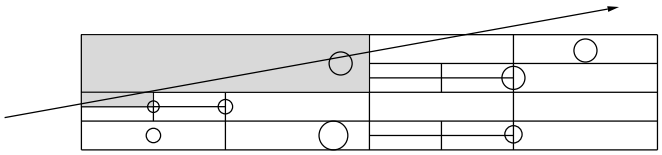


Fig. 8. 2D octree equivalent. A ray pierces the area.

2) Errors: In the top graph in figure 9(a) the individual speed-ups for BFC and DZ are shown. The overall speed-up using all 3 techniques is shown at the top of figure 9(b). The bottom 2 graphs are the mean absolute errors. The error is calculated using equation 1

$$\mathrm{m_{ae}} \;=\; \frac{1}{\mathrm{N}} \sum_{\mathrm{i}}^{\mathrm{N}} |\mathrm{x_i} - \mathrm{y_i}| \qquad (1)$$

where $\mathrm{x_i}$ and $\mathrm{y_i}$ are the total power at a far field point obtained using the optimisations and the original software respectively. The units of $\mathrm{x_i}$ and $\mathrm{y_i}$ and the mean absolute error are dB. N is the number of far field points.

The average error for BFC is 1.6×10^{-2} dB and for DZ it is 1.79×10^{-2} dB. The overall error is on average 5.55×10^{-2} dB. Examining the graphs shows no relationship between the speed-up and the magnitude of the error. Models 3 and 6 both have a higher speed-up than model 8 but the former models have a lower mean absolute error.

TABLE III
RUN TIMES OF SHADOW ALGORITHMS. SPEED-UP IS $\frac{\text{original time}}{\text{optimised time}}$

Model	**1**	**2**	**3**	**4**
Components	3	6	14	25
Original Time (sec)	6.91	13.44	46.84	62.92
Octree Time (sec)	6.87	12.22	39.39	31.96
Speed-up	1.01	1.10	1.19	1.97
Model	**5**	**6**	**7**	**8**
Components	35	49	98	147
Original Time (sec)	271.85	402.15	1177.13	2969.20
Octree Time (sec)	99.61	121.74	219.73	374.32
Speed-up	2.73	3.30	5.36	7.93

The bottom graph in figure 9(b) shows an upward trend for the error as the model size increases. The errors are attributed to the geometry of the model. If more components are added to a model there tend to be more structures which cause the errors. The error varies with the geometry and the size of the model.

V. SHADOW OPTIMISATIONS

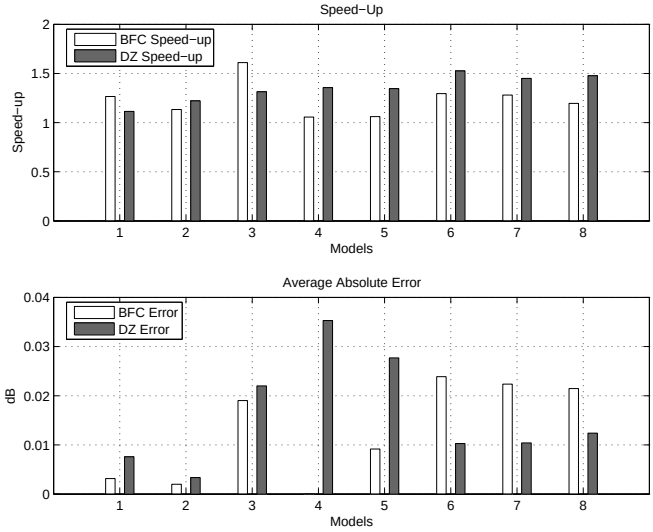
The shadow optimisations are based on an octree [7], [15], [24], [25] which is a non-linear division of a volume. Similar to the 2D grid of a road map, an octree is an indexed 3D grid of a volume. The octree quickly identifies the components which lie in the vicinity of a propagation path.

Figure 8 shows a 2D equivalent of an octree. The grid non-uniformly divides the plane into 4 smaller rectangles so that each rectangle contains only a single round object. A region of the plane with many objects is divided finer than a region with few objects. In SuperNEC-UTD an octree is built by recursively dividing the volume of a model into 8 sub-volumes. When determining which objects a ray intersects, only the objects in the rectangles intersected by the ray are inspected. In the figure only 2 rectangles are inspected before the ray intersects an object. The number of intersection tests per ray is independent of the total number of components because only a subset of the total number of components are inspected.

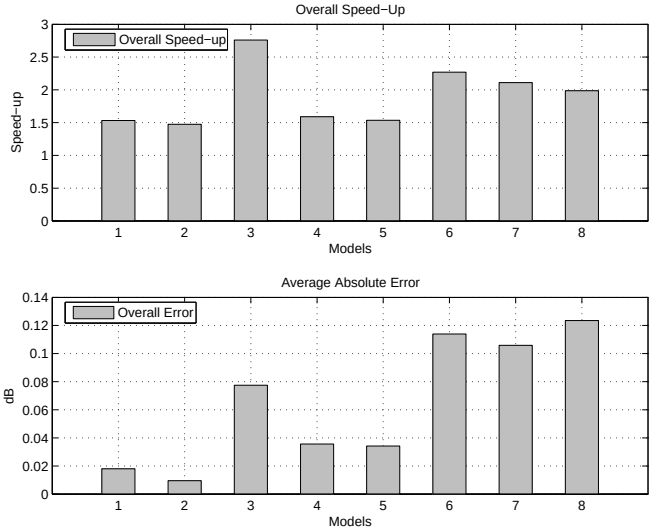
A. Results

Eight models are used to compare the original and optimised shadow algorithms. Table III shows the number of components in each model. The shadow time to calculate a 3D radiation pattern for each model is listed in the table for the original and optimised programs. The speed-up factor is given below the shadow times. In figure 10 the shadow time divided by the number of components is shown. The time is divided by the number of components to determine the scalability of the algorithm. The run time of the original algorithm, as indicated by the trend line, increases quadratically with the number of components, whereas the optimised algorithm remains fairly constants because the duration of the shadow tests is independent of the model's size.

The speed-up using the octree becomes larger as model size increases because the run time of the original program increases exponentially whereas the optimised run time increase linearly. The first model with only 3 components has a speed-up of unity. For model 8, the original shadow tests take about 50 minutes. The optimised shadow tests run for about 6 minutes. This is approximately 8 times faster.



(a) Speed-up and errors for BFC and DZ.



(b) Overall speed-up and overall errors.

Fig. 9. Speed-up and errors for test models.

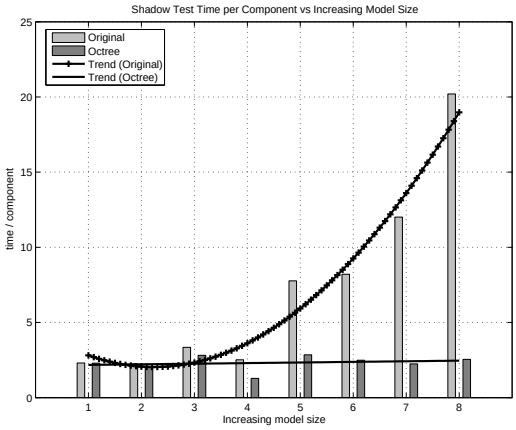


Fig. 10. Shadow time per component.

TABLE IV
MODELS FOR OVERALL RESULTS.

Model	1	2	3
Plates (total)	80	117	157
Polyhedrons	2 (38)	3(58)	5 (82)
Wedges	151 (267)	213 (378)	308 (546)
Single plates	3	3	3
Cylinders	2	8	12

VI. OVERALL RESULTS

In this section the 4 optimisations are tested collectively using 3 models. First the performance of the optimisations using simple rays is presented. The performance using simple and second order rays is presented after that. Tertiary rays are examined in [26] and are not discussed here because the results are similar to second order rays. Details of the models for both sets of tests are given in table IV. The numbers in brackets for polyhedrons are the number of plates used for those structures. For wedges the number in brackets are the total number of edges in the model.

TABLE V
TIME AND SPEED-UP RESULTS FOR SIMPLE RAYS. TIME GIVEN IN SECONDS.

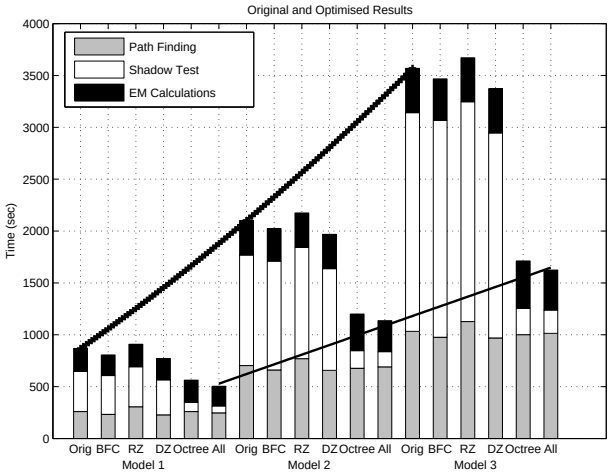
	Model 1		Model 2		Model 3	
	Time	×	**Time**	×	**Time**	×
Path finding time						
Orig	259.88		703.68		1032.99	
BFC	229.31	1.13	655.83	1.07	968.02	1.07
RZ	306.00	0.85	770.17	0.91	1127.62	0.92
DZ	228.29	1.14	656.60	1.07	969.38	1.07
All	254.88	1.02	700.60	1.00	1024.02	1.01
Shadow test time						
Orig	387.17		1064.55		2108.75	
Octree	66.83	5.79	146.75	7.25	224.97	9.37
Total run time						
Orig	868.36		2100.94		3568.11	
BFC	805.03	1.08	2022.53	1.04	3466.95	1.03
RZ	909.02	0.96	2174.59	0.97	3670.83	0.97
DZ	771.45	1.13	1967.94	1.07	3373.03	1.06
Octree	562.61	1.54	1199.44	1.75	1711.38	2.08
All	504.25	1.72	1136.49	1.85	1624.28	2.20

A. Simple Rays

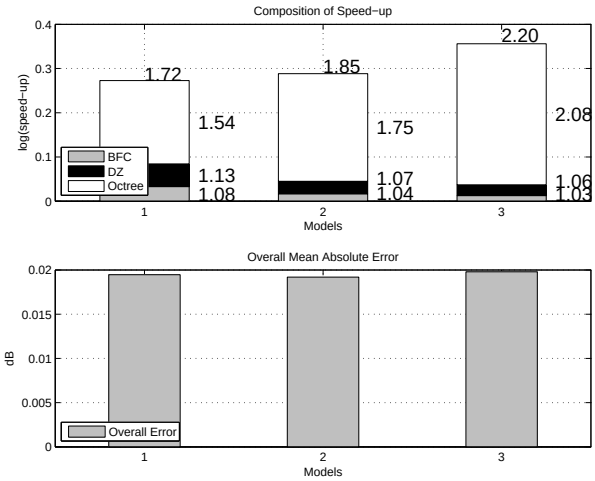
Simple rays are line of sight rays and single reflections and diffractions. For each model a 3D radiation pattern is calculated.

In table V the path finding, shadow test and overall run times are shown for the original program (orig), the 4 optimisations run individually and the run time using all optimisations combined. The speed-up factors are given in column ×. Figure 11(a) shows a graphical representation of the data in table V. The 3 bar groups represent the results for each of the 3 models. The bars are the run time for the original program, for the 4 optimisations and for the combination. The bars are divided into the path finding time, shadow time and the time for the electromagnetic calculations. Trend lines indicate the increase in run time of the original and the optimised programs.

The top graph in figure 11(b) shows the contribution of each optimisation technique to the overall speed-up. The product of the individual speed-up factors approximately equals the total



(a) Run time.



(b) Decomposition of speed-up (top graph) and overall error (bottom graph).

Fig. 11. Run time, speed-up and error graphs for simple rays.

speed-up,

$$\mathrm{S_{all}} \;\approx\; \mathrm{S_{BFC}} \cdot \mathrm{S_{RZ}} \cdot \mathrm{S_{DZ}} \cdot \mathrm{S_{octree}} \qquad (2)$$

where the S terms are the overall, BFC, RZ, DZ and octree speed-ups[1]. Taking logarithms on both sides of equation 2 yields

$$\log \mathrm{S_{all}} \;\approx\; \log \mathrm{S_{BFC}} + \log \mathrm{S_{RZ}} + \log \mathrm{S_{DZ}} + \log \mathrm{S_{octree}} \qquad (3)$$

This equation is shown in figure 11(b) (top graph) for each model. The individual speed-ups are shown to the right of each bar and the overall speed-up is given on the top of each bar.

The shadow tests in the original program take up the majority of the run time (between 45 % and 60 %). Path finding optimisations do not yield a significant reduction because the search space is too small for the optimisations to be effective. The reflection zones slightly increase the run time because of the overhead in creating and accessing the RZ. In the top graph in figure 11(b), the results for RZ are therefore not shown. BFC and DZ reduce the run time collectively by about 12 % on average. The overall speed-up factor for path finding is approximately unity.

The octree on the other hand significantly reduces the shadow time by a factor between 5 and 9 which is more than 80 % less than the original shadow time. This is clearly shown by the large reduction in the shadow time component in figure 11(a). The octree reduces the total run time by 43 % on average. Using all optimisations, the average reduction in the run time for simple rays is 48 %.

The speed-up and error for the 3 models are compared in figure 11(b). The top graph shows the speed-up for the models and the bottom the mean absolute error due to BFC and DZ. The errors are below 0.02 dB. The errors are caused by the geometry of the model and are not related to the speed-up. The speed-up increases for the models but the error shows no trend.

[1]The left and right hand sides in equation 2 are only approximately equal because the tests are run on a multitasking operating system and background processes influence the run time. When a program is run under the same conditions the time varies slightly and the product of the individual speed-ups does not exactly equal the overall speed-up.

TABLE VI

TIME AND SPEED-UP RESULTS FOR SIMPLE AND SECOND ORDER RAYS. TIME GIVEN IN SECONDS.

	Model 1		Model 2		Model 3	
	Time	×	**Time**	×	**Time**	×
Path finding time						
Orig	2768.37		8681.64		18625.46	
BFC	2272.67	1.22	7424.07	1.17	16409.18	1.14
RZ	2612.86	1.06	8236.45	1.05	17852.46	1.04
DZ	1918.87	1.44	6173.90	1.41	13408.81	1.39
All	1325.71	2.09	4683.42	1.85	10612.59	1.76
Shadow test time						
Orig	217.31		419.63		767.14	
Octree	33.43	6.50	48.99	8.57	72.20	10.63
Total run time						
Orig	3070.86		9252.55		19649.33	
BFC	2553.00	1.20	7941.59	1.17	17350.71	1.13
RZ	2899.73	1.06	8771.81	1.05	18803.91	1.04
DZ	2172.42	1.41	6643.16	1.39	14248.60	1.38
Octree	2898.95	1.06	8945.44	1.03	19084.31	1.03
All	1409.58	2.18	4815.58	1.92	10789.41	1.82

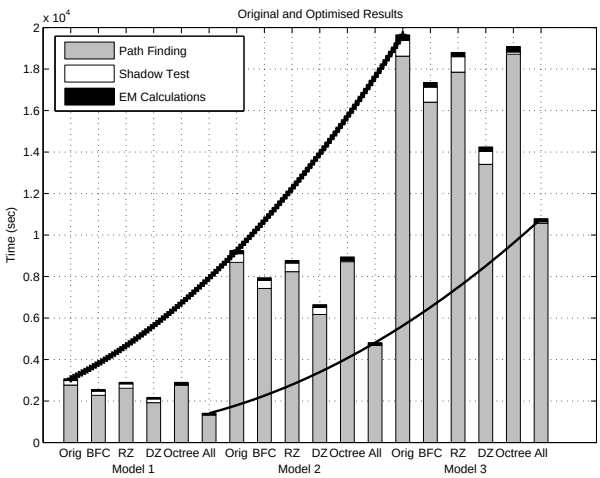
B. Simple and Second Order Rays

Three 2D radiation patterns are calculated for the 3 models to test simple and second order rays. The run times and the corresponding performance improvements are shown in table VI. Figure 12(a) shows the run times for the 3 models and in the top graph in figure 12(b) the contribution of each technique to the overall speed-up is shown.

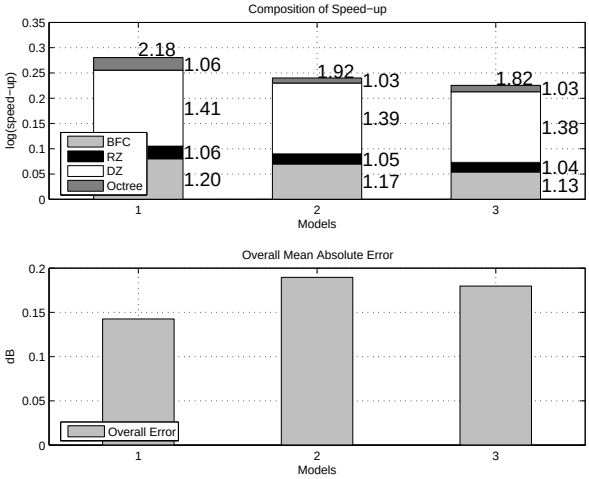
The relative importance of path finding and the shadow tests has shifted because of the large search space associated with second order rays. Path finding takes up 93 % and the shadow tests only 5 % of the run time on average in the original program. For model 3 this means the path finding time is about 5 hours and the shadow time is about 12 minutes in the original program.

The path finding optimisations reduce the original path finding time by 47 % on average. The shadow time is reduced by 88 % on average which is similar to the results for simple rays. Figure 12(a) shows that the reduction in shadow time is insignificant. Path finding optimisations provide the most significant reduction.

Figure 12(b) (top graph) shows that BFC and DZ optimisations have the largest speed-up factors, reducing the run time



(a) Run time.



(b) Decomposition of speed-up (top graph) and overall errors (bottom graph).

Fig. 12. Run time, speed-up and error graphs for simple and second order rays.

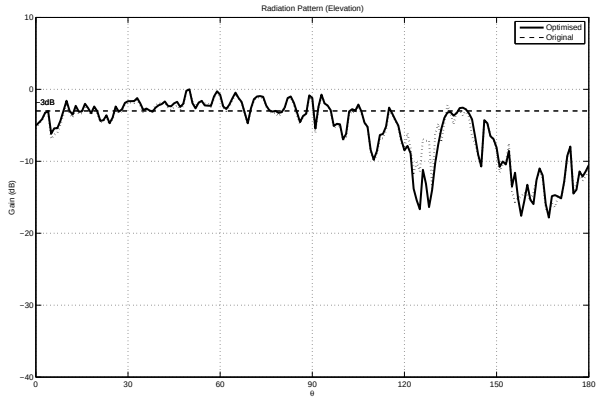


Fig. 13. Radiation pattern ($\theta = 0°..180°,\ \phi = 131°$) for model 1 using all optimisations with simple and second order rays. $\mathrm{m_{ae}}$=3.94×10^{-1} dB

respectively by $14\ \%$ and $28\ \%$ on average. RZ and the octree contribute minimally to the speed-up and reduce the run time respectively by $5\ \%$ and $4\ \%$ on average. The overall speed-up is on average 1.97 which is a $49\ \%$ reduction in the original run times. For model 3 this means the run time of 5 hours and 25 minutes is reduced to under 3 hours.

Figure 12(b) compares the speed-up (top graph) and the mean absolute error error (bottom graph). No relationship exists between the speed-up which decreases and the error which varies with the geometry of the models.

To demonstrate the accuracy of the optimisations, figure 13 compares the radiation pattern for model 1 of the original program (dotted line) with the optimised program (solid line). The shape of the original and optimised graphs are in good agreement except around $\theta = 120°$ where the errors attributed to BFC and DZ occur. The mean absolute error for this particular radiation pattern is $\mathrm{m_{ae}}$=3.94×10^{-1} dB.

VII. Discussion

This section discusses how well the solutions fulfill the scalability requirements. In the previous sections, the results of experiments showed that the geometric optimisations from ray-tracing do speed-up the software. However, the aim of the optimisations, namely the scalability of path finding and shadow tests with model size, is only partially met by the techniques.

Path finding optimisations are not a good solution. Three complex techniques are used to optimise a limited number of ray-types and a limited number of geometries. All tested models consist mostly of plates. Adding more cylinders will decrease the speed-up. The techniques introduce errors. During the tests the errors where small and negligible but they increase with model size.

The major problem with the techniques is their un-scalability with model size. The experiments showed that the run time when using the optimisations is reduced in comparison with the original program but the run time continues to increase quadratically (see figure 12(a)) and consequently that the speed-up decreases with model size (see figure 12(b) which shows a downward trend in the speed-up). The aim of the optimisations is not met by the path finding optimisations. Simply using geometric optimisations to reduce the search space for second and higher order rays has not lead to a scalable solution.

This contrasts with the octree optimisation which firstly uses a single concept (the octree) that is simple to implement, generally applicable to any geometry and ray-type and does not introduce errors. Secondly, the technique leads to shadow tests which scale with model size and therefore meet the requirements.

Looking at the un-scalability of the path finding optimisations objectively, one must remember that the aim of these techniques is to optimised second and higher order rays. This confines the problem to only those rays. Direct and first order rays are optimised in a scalable fashion using the octree. For these rays the shadow time dominates and the path finding optimisations can be disabled which mean no errors are introduced. First order effects provide a good initial radiation pattern sufficient for engineering purposes for most structures [27] and which can be used to quickly determine solutions during problem optimisation. Second and higher order rays are less significant than first order effects and the accuracy of the UTD diffraction solutions becomes questionable for higher order rays [28].

Figure 14 shows the radiation pattern in the elevation plane

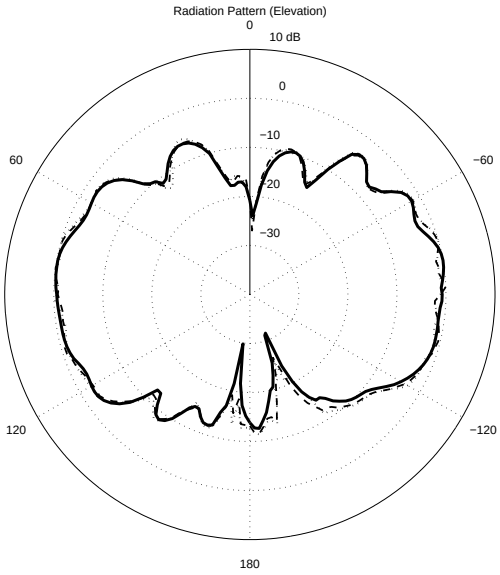


Fig. 14. Radiation pattern ($\phi = 0°,\ \theta = 0°..360°$) for a dipole in the presence of 2 six sided boxes and 2 single plates.

of a dipole in the presence of 2 six sided boxes and 2 single plates. The bold line is for simple rays. When second order rays are added the pattern changes to the dotted line and when third order rays are added the pattern changes to the dashed line. In the figure, higher order rays change the pattern around, eg. $\phi = 180°$ but do not influence the overall shape which is primarily determined by first order effects.

VIII. Conclusion

This paper has highlighted the problems in the brute force UTD implementation in SuperNEC. The high run times are caused by the brute force path finding and shadow test algorithms which do not scale. An ideal characteristic of any optimisation is that it must scale with model size. Four geometric optimisations where presented. These included 3 techniques to optimise the path finding algorithm by determining in advance which components and spatial regions are illuminated by rays. A further technique optimised the shadow algorithm by reducing the number of intersections to only a small subset of the total number of objects in a model. Path finding optimisations do not scale with model size and are limited to certain geometries and ray-types. The shadow optimisations do scale with model size and work with all geometrical structures and all ray-types.

At present the path finding optimisations only consider the geometric properties of the rays. Future research could investigate the use of the electromagnetic fields propagated by the rays, to quickly eliminate those rays whose electromagnetic fields have attenuated below a certain threshold. This technique can then be used to improve the scalability of path finding.

References

[1] R. G. Kouyoumjian and P. H. Pathak, "A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface," *Proceedings of the IEEE*, vol. 62, no. 1, Nov. 1974.

[2] J. Keller, "Geometrical theory of diffraction," *Journal of the Optical Society of America*, vol. 52, no. 2, pp. 116–130, 1962.

[3] G. A. Deschamps, "Ray techniques in electromagnetics," *Proceedings of the IEEE*, vol. 60, no. 9, Sept. 1972.

[4] A. Fourie and D. Nitch, "SuperNEC: Antenna and indoor propagation simulation program," *IEEE Antennas and Propagation Magazine*, vol. 42, no. 3, June 2000.

[5] R. F. Harrington, *Field Computation By Moment Methods*. New York: Macmillan, 1968.

[6] R. Hartleb, "Optimised ray tracing for the SuperNEC implementation of the uniform theory of diffraction," MSc(Eng) Dissertation, University of the Witwatersrand, Johannesburg, 2006, Appendix A: Analysis of the run time of SuperNEC-UTD: a MoM electromagnetic simulation package.

[7] A. S. Glassner, Ed., *An Introduction to Ray Tracing*. Academic Press, 1989.

[8] O. Gutiérrez, F. S. de Adana, I. González, J. Pérez, and M. F. Catedra, "Ray-tracing techniques for mobile communications," *Applied Computational Electromagnetics Society (ACES) Journal*, vol. 15, no. 3, pp. 209–231, Nov. 2000.

[9] A. Formella, F. A. Agelet, and J. M. H. Rábanos. (1999, Sept.) Acceleration techniques for ray-path searching in urban and suburban environments to implement efficient radio propagation simulators. Last accessed 27 May 2006. [Online]. Available: http://trevinca.ei.uvigo.es/˜formella/inv/tel/formella-2002-acceleration.pdf

[10] J. Perez, J. A. Saiz, O. M. Conde, R. P. Torre, and M. F. Catedra, "Analysis of antennas on board arbitrary structures modeled by NURB surfaces," *IEEE Transactions on Antennas and Propagation*, vol. 45, no. 6, June 1997.

[11] S. J. Fortune, D. M. Gay, B. W. Kernighan, O. Landron, R. A. Valenzuela, and M. H. Wright, "Wise design of indoor wireless systems: Practical computation and optimization," *IEEE Computational Science and Engineering*, 1995, spring.

[12] J. W. McKown and R. L. Hamilton, "Ray tracing as a design tool for radio network," *IEEE Network Magazine*, vol. 5, pp. 27–31, Nov. 1991.

[13] M. F. Catedra, J. Pérez, F. S. de Adana, O. Gutiérrez, J. Cantalapiedra, and I. González, "Efficient ray-tracing techniques for three-dimensional analyses of propagation in mobile communications: Application to picocell and microcell scenarios," *IEEE Antennas and Propagation Magazine*, vol. 40, no. 2, pp. 15–28, Apr. 1998.

[14] D. Didascalou, "Ray density normalisation for ray-optical wave propagation modelling in arbitrarily shaped tunnels," *IEEE Transactions on Antennas and Propagation*, vol. 48, no. 9, pp. 1316–1325, Sept. 2000.

[15] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice*. Addison-Wesley, 1990.

[16] M. Kimpe, H. Leib, O. Maquelin, and T. H. Szymanski, "Fast computational techniques for indoor radio channel estimation," *Computing in Engineering and Science*, vol. 1, no. 1, pp. 31–41, Jan–Feb 1999.

[17] F. A. Agelet, F. P. Fontán, and A. Formella, "Fast ray-tracing for microcellular and indoor environments," *IEEE Transactions of Magnetics*, vol. 33, no. 2, Mar. 1997.

[18] G. E. Athanasiadou, A. Nix, and J. McGeehan, "A microcellular ray-tracing propagation model and evaluation of its narrowband and wideband predictions," *IEEE Journal on Selected Areas in Communication*, vol. 18, no. 3, Mar. 2000.

[19] A. Glassner, "Space subdivision for fast ray tracing," *IEEE Computer Graphics and Applications*, vol. 4, no. 10, pp. 15–22, Oct. 1984.

[20] I. E. Sutherland, R. F. Sproull, and R. A. Schumacker, "A characterization of ten hidden-surface algorithms," *ACM Computing Surveys*, vol. 6, no. 1, pp. 1–5, Mar. 1974.

[21] R. Hartleb, "Optimised ray tracing for the SuperNEC implementation of the uniform theory of diffraction," MSc(Eng) Dissertation, University of the Witwatersrand, Johannesburg, 2006, Appendix B: Path Finding Optimisations for SuperNEC-UTD.

[22] ——, "Optimised ray tracing for the SuperNEC implementation of the uniform theory of diffraction," MSc(Eng) Dissertation, University of the Witwatersrand, Johannesburg, 2006, Appendix C: Shadow Test Optimisations for SuperNEC-UTD.

[23] D. A. McNamara, C. W. I. Pistorius, and J. A. G. Malherbe, *Introduction to the Uniform Theory of Diffraction*. Artech House, 1990.

[24] H. Samet, *The design and analysis of spatial data structures*. Addison-Wesley, 1990.

[25] ——, "The quadtree and related hierarchical data structures," *ACM Computing Survey*, vol. 16, no. 2, pp. 187–260, June 1984.

[26] R. Hartleb, "Optimised ray tracing for the SuperNEC implementation of the uniform theory of diffraction," MSc(Eng) Dissertation, University of the Witwatersrand, Johannesburg, 2006, Appendix D: Results of Geometric Optimisations used collectively in SuperNEC-UTD.

[27] R. J. Marhefka and W. D. Burnside, "Numerical Electromagnetic Code (NEC) - Basic Scattering Code Part I: User's Manual," ElectroScience Laboratory, The Ohio State University, Columbus, Ohio 43212, Tech. Rep. 784508-18, 1979.

[28] W. D. Burnside and R. J. Marhefka, "Antenna applications," in *Antenna Handbook*, Y. T. Lo and S. W. Lee, Eds. New York: Chapman & Hall, 1993, vol. 3, ch. 20.

Part II

Appendices

Appendix A

Analysis of SuperNEC UTD: a MoM electromagnetic simulation package.

Abstract

Shortcomings are identified in the ray-tracing code of SuperNEC-UTD which is a numerical electromagnetic software simulator. The shortcomings are the ray tracing code which consists of the path finding and shadow test algorithms. A brief overview of SuperNEC-UTD is given which focuses on the ray-tracing software. Tests models are used to determine which sections of the ray-tracing code are slow. Ray-tracing is slow in SuperNEC-UTD because both the path finding and shadow tests use a brute force algorithm. Path finding examines all possible combinations of geometric ray paths for a given pair of source and observer points without regard to the geometric layout of the model. The shadow tests compare a ray against every component in a model to determine if the ray intersects a component. As model size increases the run times of both algorithms grow exponentially and do not scale. Depending on the ray-types different sections of SuperNEC need to be optimised. For simple rays (line of sight, single reflections and single diffractions), only the shadow tests need to be optimised. For simple rays the shadow tests take up between 40 % and 50 % and the path finding between 30 % and 40 % of the total run time. When complex rays (simple rays and all higher order rays) are used, the path finding time takes up the majority of the run time (approximately 90 %) while the shadow tests take up less the 10 % of the run time.

A.1 Introduction

SuperNEC [1, 2] is a numerical electromagnetic simulator that includes an implementation of the uniform geometric theory of diffraction (UTD) [3, 4]. SuperNEC-UTD uses ray-tracing to determine the paths of high frequency electromagnetic waves. In practice this code is too slow and needs to be optimised.

The purpose of this report is to clearly identify which sections of the UTD code are slow and why those parts of the software are slow. This report only analyses the existing software and does not present any solutions.

The UTD code consists of 3 parts: path finding, shadow tests and electromagnetic field calculations. The first 2 parts are the ray-tracing code and the focus of this report. Analysis of the ray-tracing code shows that both path finding and the shadow tests are slow because of the brute force algorithms which are employed. Path finding exhaustively examines all combinations of components in a model to find a valid geometrical path. The shadow test algorithm blindly compares a ray against each component in a model to determine if the ray intersects a component. The run time of both algorithms increases exponentially with increasing model size.

In section A.2 an overview of SuperNEC-UTD is given together with an explanation of the ray-tracing code. Following that, in section A.3, the bottlenecks in the code are identified. Section A.4 presents the reasons why the path finding and the shadow test algorithms are slow

A.2 An Overview of SuperNEC

A.2.1 Description of SuperNEC-UTD

SuperNEC-UTD uses ray methods to find the paths of high frequency electromagnetic waves and to determine the electromagnetic field at an observer due to a radiating source in the presence of a structure. Using the method of moments (MoM) [2, 5], structures and antennas are modelled using a wire-frame of segments. In general many segments are used, to construct a complex antenna or structure. Instead of using a mesh of segments to build up models, in SuperNEC-UTD models consist of plates and cylinders (collectively referred to as components).

To determine the radiation pattern of an antenna in the presence of a structure, rays are traced from the individual segments of the antenna. Rays can reflect and diffract off the components while propagating to points in the far field. UTD is used to determine the electromagnetic field at any point along the rays.

SuperNEC-UTD consists of three parts: path finding, shadow tests and field calculations. Figure A.1 shows how these 3 parts are linked. Once a valid geometric path has been found

Figure A.1: Flow diagram of UTD in SuperNEC.

by path finding, the shadow tests are run to determine if a component blocks that path. Once it has been determined that the path is not shadowed, the electromagnetic (EM) field calculations are executed using the UTD. Only if the path is geometrically valid and the ray is not shadowed are the fields calculated.

A.2.2 Ray-Tracing in SuperNEC-UTD

Ray-tracing in SuperNEC is based on an interpolative method. A source and an observer location are specified. The software will attempt to fit a geometrically valid ray between

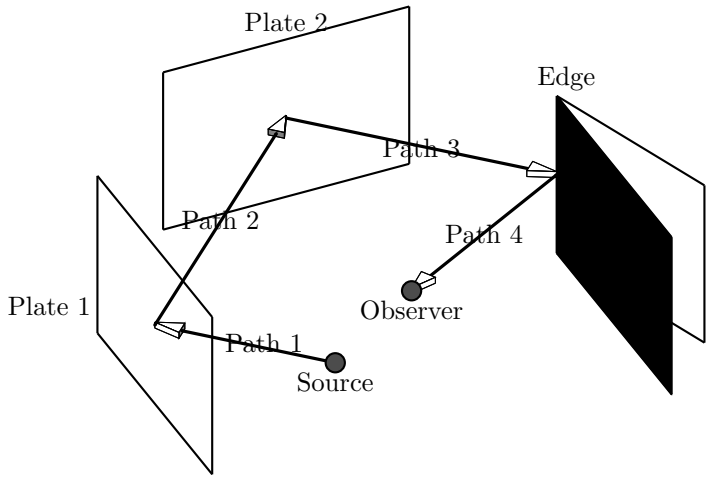


Figure A.2: An example ray consisting of 4 paths. The ray consists of a plate reflection (RP), followed by another plate reflection (RP) and an edge diffraction (DE).

Table A.1: Explanation of ray codes.

RP	reflection off a plate	**RC**	reflection off a cylinder	**RL**	reflection from the bottom or top rim of a cylinder
DE	diffraction from an edge of a plate	**DC**	diffraction from a cylinder	**DL**	diffraction from the (bottom or top) rim of a cylinder

the source and observer. A ray can reflect in or diffract off objects when travelling from the source to the observer.

Figure A.2 illustrates this procedure. A ray starts at the source, propagates to the first plate where it reflects to a second plate. Then the ray reflects towards the edge where it is diffracted to the observer. A ray consists of one or more paths. In figure A.2, the ray consists of 4 paths. Each path lies between one of the components.

The ray shown in figure A.2, is a geometrically valid ray because it obeys the laws of reflection [6] and diffraction [4]. The ray could also consist of fewer paths and one or more of the plates could be replaced by cylinders.

A class hierarchy in SuperNEC-UTD determines which rays can be shot. Figure A.3 shows the ray-types supported in SuperNEC. The categories first, second and higher order rays are subdivided according to the components they interact with. The notation used in figure A.3

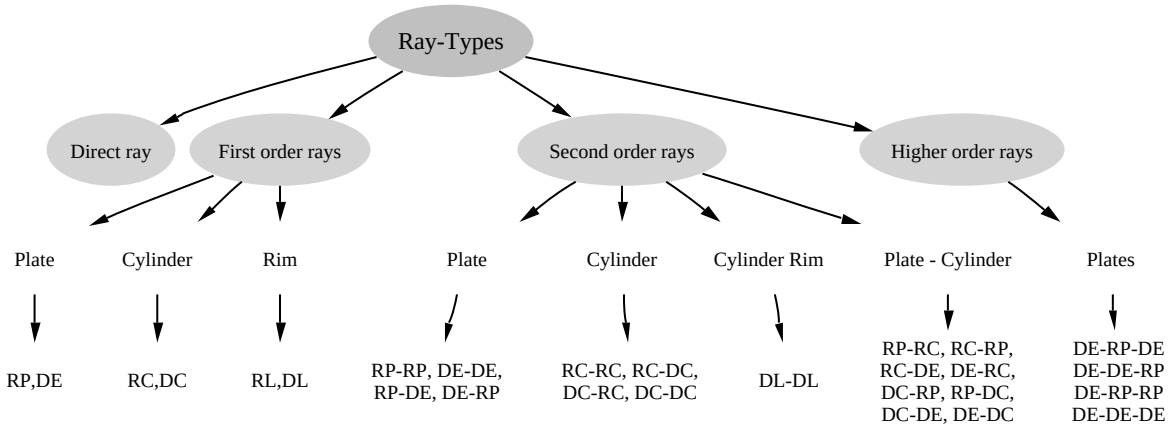


Figure A.3: Hierarchy of ray-types used in SuperNEC-UTD.

is explained in table A.1. Second and higher order rays are indicated by combinations of the codes listed in the table. Two reflections off a plate followed by an edge diffraction (as

in figure A.2) would be indicated by RP-RP-DE.

The next 2 sections explain in detail the two ray-tracing parts: path-finding and shadow tests.

Path Finding

Path finding uses a brute force algorithm to determine geometrically valid rays from a source to an observer. Given a source and an observer location and a particular ray type, path finding will find all geometrically valid rays between the source and observer.

When a model consists of only one plate, there can only be one single plate reflection ray between a source and an observer. For a model with 2 plates there are at most 2 single plate reflection rays between a fixed source and a fixed observer. For the same model, there are at least 6 different single edge diffraction rays (a plate has at least 3 edges, so for 2 plates there are at least 6 edges). The number of combinations of ray paths in a model is referred to as the *search space*. In a model with 2 plates and therefore at least 6 edges, path finding will search (the search space) for an edge diffraction ray along each of the 6 edges. For higher order rays and models with more components the number of permutations is correspondingly higher.

In figure A.2 a RP-RP-DE ray is shown. In path finding the software will search for valid rays using all (arbitrary) combinations of plates 1 to 3, eg. another RP-RP-DE ray could be from plate 2, to plate 3 and to an edge of plate 1. Determining the points of reflection and diffraction is done using either an iterative method or using image theory.

For path finding to find a geometrically valid ray, the ray must not only obey the laws of reflection and diffraction for the different components, but the ray must also reflect and diffract off the finite sized components. The components have finite surface area (in the case of plates and cylinders) and finite length (in the case of edges). The path finding algorithm first assumes infinitely sized components, and after finding a geometrically valid ray checks if the ray is on the finite sized components.

Shadow Tests

Shadow tests determine if a geometrically valid ray from path finding is blocked by any components in a model. Figure A.4, shows the same ray as in figure A.2. The ray is now block by a fourth component. The shadow test detects such intersections and declares a ray shadowed. A shadowed ray is removed from further processing.

The shadow test algorithm is a naive brute force method. Except for the components involved in the ray paths, ie. plates 1 and 2 and the edge in figure A.2, the algorithm checks each ray-path against each component, one at a time, to see if they block the ray path.

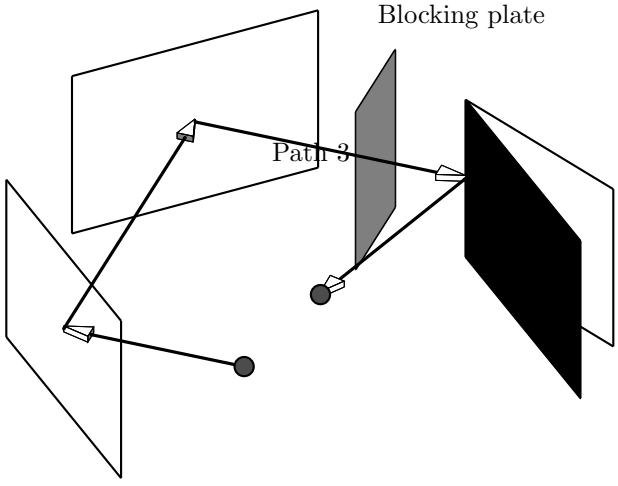


Figure A.4: An example ray that is shadowed.

Electromagnetic Field Calculations

Once a geometrically valid ray path is found that is not shadowed, UTD is used to propagated the electromagnetic field from the source, along the ray path to the observer. Determining the electromagnetic field along a ray is mathematically difficult. Calculating the UTD coefficients is a computationally expensive operation, involving complex mathematics, integrals and iterative methods [6]. The focus of this research is to optimise the ray-tracing in SuperNEC-UTD and the electromagnetic field calculations are not optimised.

A.3 Analysis of SuperNEC-UTD

In this section the results of tests are discussed that pinpoint the bottlenecks in SuperNEC-UTD. The run-time of the path finding, shadow test and electromagnetic parts (as shown in figure A.1) is recorded. Also, the number of rays shot is recorded.

Three tests are run using different ray-types. In the first test, only simple rays are used. In the second test simple and second order rays are used. Tertiary rays are disabled because they take a long time to run. Finally in a third test for tertiary rays a smaller model is used to keep the run time reasonable.

This section starts off by giving an overview of the tests. This includes the hardware and software used for the tests. Also information on the models and the ray-types is provided. The empirical results are presented in section A.3.2. Section A.3.3 analyses the results of the tests.

A.3.1 Overview of the Tests

Table A.2: Hardware and software for the tests.

Hardware	
Processor	Intel(R) Pentium(R) 4 CPU [7]
Processor Frequency	3.80 GHz
Random Access Memory (RAM)	1.00 GB
Software	
Operating System (OS)	Microsoft Windows XP Professional
OS Version	Version 2002, Service Pack 2
SuperNEC Version	2.9
SuperNEC Compile Configuration	WIN32 Release
Compiler	Microsoft Visual C++ 6.0

Hardware and software for the tests. The tests are run using the hardware and software listed in table A.2. SuperNEC compile configuration in the 2nd last line of the table indicates that the compiler optimisations (for Intel Pentium 4 processors [7]) are used and debugging is disabled.

Ray Types Table A.3 lists the 3 groups of ray-types: simple, combined and tertiary rays. The simple ray-types include direct rays (ie. line of sight) and rays with only a single coupling mechanism ie. single reflections off plates, cylinders and end caps (the tops and bottoms of cylinders) and single diffractions at plate edges, around cylinders and around end caps. The combined ray-types include the simple rays and all second order rays. Second order rays consist of 2 arbitrary combinations of reflections and diffractions. In the third group all tertiary rays are classed. Tertiary rays have 3 arbitrary combinations of reflections and diffractions.

Test Models Table A.4 lists the two test models that are used during the experiments. The first model in table A.4 is used with the simple and combined ray types listed in table A.3. For tertiary rays the second model in table A.4 is used. The percentage behind the components is the percentage of the total number of components.

Table A.3: Ray-types for the tests.

Simple Ray Class			
Component	**Primary**	**Secondary**	**Tertiary**
Line of sight	direct ray		
Plate	RP, DE	–	–
Cylinder	RC, DC	–	
End cap	RL, DL	–	

Combined Ray Class			
Component	**Primary**	**Secondary**	**Tertiary**
Line of sight	direct ray		
Plate	RP, DE	RP-RP, RP-DE, RP-RC, RP-DC, DE-DE, DE-RP, DE-RC, DE-DC	–
Cylinder	RC, DC	DC-DC	
End cap	RL, DL	DL-DL	

Tertiary Ray Class			
Component	**Primary**	**Secondary**	**Tertiary**
Plate	–	–	RP-RP-RP, DE-DE-DE, DE-DE-RP, DE-RP-DE, DE-RP-RP

Table A.4: Test models for the experiments.

	Simple and combined	**Tertiary**
Simulation Frequency (MHz)	300	300
Number of segments	7	7
Number of plates	99 (89.0 %)	20 (100.0 %)
Number of cylinders	12 (11.0 %)	0 (0.0 %)

The radiation patterns that are calculated during the experiments are listed in table A.5. For the simple rays a full 3 dimensional radiation pattern with 1 degree increments in both the azimuth and elevation planes is determined. Using the combined ray class a 3 dimensional radiation pattern with 4 degree increments in the azimuth and elevation planes is calculated. For the tertiary rays three 2 dimensional radiation patterns are calculated.

A.3.2 Empirical Results

Empirical Results for Simple Rays

The number of rays traced using simple rays is shown in table A.6. The table is divided into the total number of rays traced, the geometrically valid rays and those rays that are not shadowed. The percentages under the total number of rays are the percentage each ray type makes up of the total number of rays, eg. 26.54 % of all rays traced are RP rays. For the geometrically valid rays, the percentage of the total number of rays that are geometrically

Table A.5: Radiation patterns used with the test models. The column *degrees* gives the number of sample points taken in the azimuth or in the elevation planes.

Simple Ray Tests						
	θ			ϕ		
Plane	Start	End	Degrees	Start	End	Degrees
3D	0.0°	180.0°	181	0.0°	360.0°	361

Combined Ray Tests						
	θ			ϕ		
Plane	Start	End	Degrees	Start	End	Degrees
3D	0.0°	180.0°	46	0.0°	360.0°	91

Tertiary Ray Tests						
	θ			ϕ		
Plane	Start	End	Degrees	Start	End	Degree Degrees
XY Plane	90.0°	90.0°	1	0.0°	360.0°	181
XZ Plane	0.0°	360.0°	181	0.0°	0.0°	1
YZ Plane	0.0°	360.0°	181	90.0°	90.0°	1

Table A.6: Number of simple rays traced.

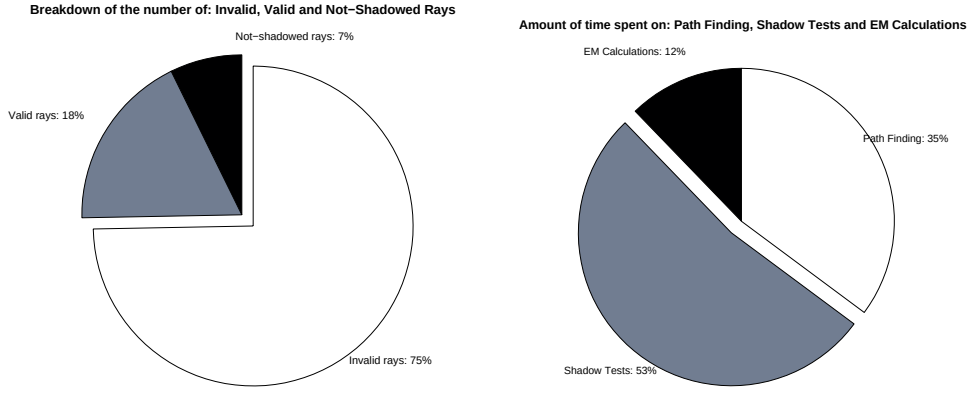
Ray-Types	Total Number of Rays		Geometrically Valid Rays		Not Shadowed Rays	
	Count	%	Count	%	Count	%
Direct	457436	0.27%	457436	100.00%	124725	27.27%
RP	45286164	26.54%	167685	0.37%	49620	29.59%
DE	75476940	44.24%	8550558	11.33%	2867820	33.54%
RC	5489232	3.22%	1111382	20.25%	725170	65.25%
DC	10978464	6.43%	911170	8.30%	647806	71.10%
REC	10978464	6.43%	4085	0.04%	1804	44.16%
DEC	21956862	12.87%	21956798	100.00%	8966792	40.84%
Totals	**170623562**	**100.00%**	**33159114**	**19.43%**	**13383737**	**40.36%**

valid is given in column 5, eg. 0.37 % of the 45286164 RP rays are geometrically valid. The percentages of the geometrically valid rays that are not shadowed is shown in the last column, eg. 29.59 % of geometrically valid RP rays are not shadowed.

The breakdown of the time spent for each ray on path finding, shadow tests and EM calculations is shown in table A.7. The total times are shown in the last row. The percent of run time used by each ray-type is shown next to the individual times. The percentages in the last row are the percent each section takes up of the total run time. For example, 10.56 % of path finding time is spent looking for DE rays. Then, 19.62 % of the time spent on the shadow tests is used to determine which of the DE rays are shadowed. Finally, 33.18 % of the time spent calculating the electromagnetic fields, is used to determine the DE fields. The overhead in the last row is the time to initialise the program

Table A.7: Ray tracing time for simple rays.

Ray-Types	Path Finding		Shadow Tests		EM Calculations	
	Time (sec)	%	Time (sec)	%	Time (sec)	%
Direct	0.00	0.00%	10.30	0.71%	0.53	0.16%
RP	19.63	2.02%	5.04	0.35%	0.48	0.14%
DE	102.35	10.56%	284.68	19.62%	111.84	33.18%
RC	205.04	21.15%	40.53	2.79%	7.00	2.08%
DC	55.47	5.72%	37.92	2.61%	13.54	4.02%
REC	11.11	1.15%	0.09	0.01%	0.03	0.01%
DEC	575.95	59.40%	1072.04	73.90%	203.68	60.42%
Totals	**969.56**	**34.76%**	**1450.60**	**52.01%**	**337.10**	**12.09%**
Total Run-time	**2789.23**	**Overhead**	**31.98**	**1.15%**		



(a) Breakdown of the number of: invalid, valid and not-shadowed rays.

(b) Breakdown of the time spent on: path-finding, shadow tests and EM calculations.

Figure A.5: Breakdown of number of rays and run time for simple ray-types.

The results presented in tables A.6 and A.7 are illustrated graphically in figures A.5(a) and A.5(b) respectively. The pie chart in figure A.5(a) represents the total number of rays traced. 75 % of the total number of rays are not geometrically invalid, 18 % are geometrically valid and only 7 % are not shadowed. The breakdown of the total run time is shown in figure A.5(b). Path finding time takes 35 % of the run time, the shadows 53 % and the EM

calculations 12 %.

Empirical Results for Combined Rays

The number of rays traced using the combined ray class is listed in table A.8. Each row has the same format. For example, 4842915 single edge diffraction (DE) rays are traced. These are 0.19 % of the total number of rays. Only 553965 of those DE rays are geometrically valid. Of the 553965 geometrically valid only 33.63 %, that is 186321 rays are not shadowed. The last row lists the totals of the columns. 1.90 % of the total number of rays are geometrically valid. 19.95 % of the geometrically valid rays are not shadowed.

Table A.8: Number of combined rays traced.

Ray-Types	**Total Number of Rays**		**Geometrically Valid Rays**		**Not Shadowed Rays**	
	Count	%	Count	%	Count	%
Direct	29351	0.00%	29351	100.00%	21030	71.65%
RP	2905749	0.11%	11258	0.3874%	3202	28.4420%
DE	4842915	0.19%	553965	11.4387%	186321	33.6341%
RC	352212	0.01%	71756	20.3730%	46726	65.1179%
DC	704424	0.03%	59267	8.4135%	41887	70.6751%
REC	704424	0.03%	244	0.0346%	115	47.1311%
DEC	1408782	0.05%	1408718	99.9955%	575082	40.8231%
RP-RP	284763402	11.06%	4449	0.0016%	381	8.5637%
RP-DE	474605670	18.43%	13307618	2.8039%	4668835	35.0839%
DE-RP	474605670	18.43%	1966649	0.4144%	324754	16.5131%
DE-DE	794238060	30.84%	19823618	2.4959%	2064600	10.4148%
RC-RC	3874332	0.15%	259844	6.7068%	64686	24.8942%
RC-DC	7746816	0.30%	95453	1.2322%	51821	54.2895%
DC-RC	7746816	0.30%	48775	0.6296%	12218	25.0497%
DC-DC	15493632	0.60%	57215	0.3693%	25115	43.8958%
DL-DL	1662498	0.06%	1662330	99.9899%	574174	34.5403%
RP-RC	34868988	1.35%	42762	0.1226%	12309	28.7849%
RP-DC	69737976	2.71%	31304	0.0449%	9685	30.9385%
DE-RC	58114980	2.26%	4163498	7.1642%	377303	9.0622%
DE-DC	58114980	2.26%	886319	1.5251%	246999	27.8680%
RC-RP	34868988	1.35%	6232	0.0179%	2684	43.0680%
RC-DE	58114980	2.26%	3549291	6.1074%	240984	6.7896%
DC-RP	69737976	2.71%	3770	0.0054%	1211	32.1220%
DC-DE	116202240	4.51%	805854	0.6935%	191637	23.7806%
Total	**2575445861**	**100.00%**	**48841219**	**1.8964%**	**9743759**	**19.9499%**

The time used by each ray-type is listed in table A.9. Each row has the same format. For example, for RP-DE rays, path finding takes 1013.52 sec ($\approx$ 17 minutes) which is 1.34 % of the total path finding time. The shadow tests for RP-DE rays take 1519.72 sec ($\approx$ 25 minutes) which is 55.11 % of the total shadow test time. To calculate the electromagnetic fields requires 170.52 sec ($\approx$ 3 minutes). The sums of the 3 UTD sections are shown at the bottom of the table. The percentages are relative to the total run time which is shown in the last row of the table. The overhead in the last row is the time to initialise the program.

A breakdown of the number of rays is shown in figure A.6(a). 98 % of the rays are geometrically invalid, 2 % are geometrically valid and less then 1 % of the rays are not shadowed. From the pie chart in figure A.6(b), path finding takes 96 %, the shadow tests take 4 % and the EM calculations take less than 1 % of the total run time.

Empirical Results for Tertiary Rays

The results for tertiary rays are shown in tables A.10 and A.11. Columns 2 and 3 in table A.10 show the total number of rays traced and the percentage of total rays for each ray class, with the total given at the bottom of the columns. The next 2 columns are the number of geometrically valid rays. The percentages indicate how many of the total number of rays are geometrically valid. The final 2 columns are the number of rays which are not shadowed. The percentages are the number of geometrically valid rays which are not shadowed. For example, 27797000 triple reflection (RP-RP-RP) rays are traced. This is 5.01 % of the total number of rays (555093000). Only 151 rays are geometrically valid.

Table A.9: Ray tracing time for combined rays.

Ray-Types	Path Finding		Shadow Tests		EM Calculations	
	Time (sec)	%	Time (sec)	%	Time (sec)	%
Direct	0.00	0.00%	0.566	0.02%	0.078	0.02%
RP	1.06	0.00%	0.459	0.02%	0.016	0.01%
DE	7.214	0.01%	19.698	0.71%	6.801	2.17%
RC	12.991	0.02%	2.535	0.09%	0.386	0.12%
DC	3.611	0.00%	2.554	0.09%	0.782	0.25%
REC	0.653	0.00%	0	0.00%	0	0.00%
DEC	37.471	0.05%	67.71	2.46%	13.086	4.18%
RP-RP	708.529	0.94%	0.093	0.00%	0	0.00%
RP-DE	1013.519	1.34%	1519.72	55.11%	170.52	54.47%
DE-RP	928.285	1.23%	109.859	3.98%	11.828	3.78%
DE-DE	18040.065	23.91%	669.463	24.28%	50.446	16.11%
RC-RC	2787.229	3.69%	7.451	0.27%	1.126	0.36%
RC-DC	493.702	0.65%	6.837	0.25%	1.335	0.43%
DC-RC	176.498	0.23%	2.07	0.08%	0.31	0.10%
DC-DC	87.483	0.12%	4.777	0.17%	1.242	0.40%
DL-DL	90.505	0.12%	119.846	4.35%	24.152	7.71%
RP-RC	1274.665	1.69%	1.359	0.05%	0.219	0.07%
RP-DC	334.153	0.44%	1.176	0.04%	0.293	0.09%
DE-RC	19977.733	26.48%	59.979	2.17%	8.741	2.79%
DE-DC	1532.952	2.03%	42.692	1.55%	8.438	2.70%
RC-RP	1301.785	1.73%	0.344	0.01%	0.064	0.02%
RC-DE	24626.457	32.64%	68.79	2.49%	6.453	2.06%
DC-RP	339.204	0.45%	0.283	0.01%	0	0.00%
DC-DE	1663.042	2.20%	49.49	1.79%	6.763	2.16%
Totals	**75438.81**	**96.07%**	**2757.75**	**3.51%**	**313.08**	**0.40%**
Total Run-time	**78523.46**	(Overhead 13.82 sec 0.02 %)				

Of the 151 geometrically valid rays, only 1 is not shadowed. This represents 0.66 % of the 151 geometrically valid rays.

Table A.10: Number of tertiary rays traced.

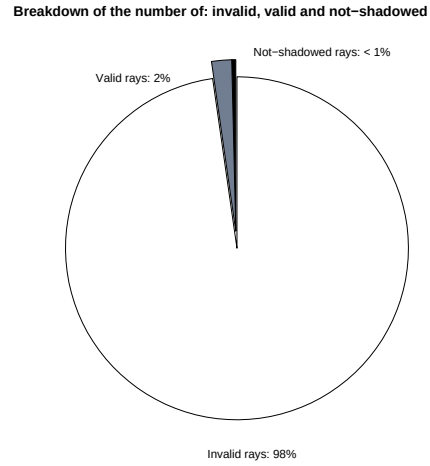
Ray-Types	Total Number of Rays		Geometrically Valid Rays		Not Shadowed Rays	
	Count	%	Count	%	Count	%
RP-RP-RP	27797000	5.01%	151	0.00%	1	0.66%
DE-RP-RP	58520000	10.54%	4174	0.01%	272	6.52%
DE-DE-RP	114114000	20.56%	72909	0.06%	18658	25.59%
DE-RP-DE	108416000	19.53%	177136	0.16%	21989	12.41%
DE-DE-DE	246246000	44.36%	5460837	2.22%	2125973	38.93%
Total	**555093000**	**100.00%**	**5715207**	**1.03%**	**2166893**	**37.91%**

The timing results in table A.11 are given in the same format. For example the path finding time for triple diffraction rays (DE-DE-DE) is 1740.68 seconds ($\approx$ 28 minutes) which is 66.60 % of the total path finding time. The shadow test for DE-DE-DE rays take 78.46 seconds which is 96.53 % of the total shadow time. The EM calculations ran for 74.38 seconds which is 98.51 % of the total time spent on all the EM calculations.

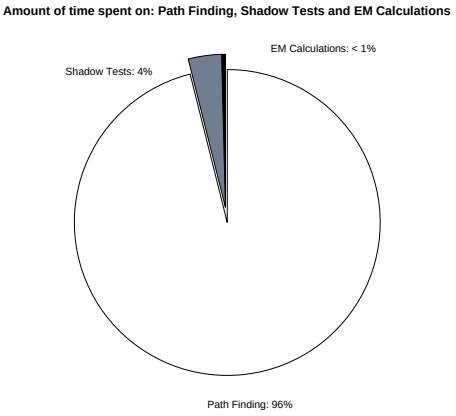
Table A.11: Ray tracing time for tertiary rays.

Ray-Types	Path Finding		Shadow Tests		EM Calculations	
	Time (sec)	%	Time (sec)	%	Time (sec)	%
RP-RP-RP	71.43	2.73%	0.00	0.00%	0.00	0.00%
DE-RP-RP	76.23	2.92%	0.03	0.04%	0.02	0.02%
DE-DE-RP	380.72	14.57%	0.96	1.19%	0.47	0.62%
DE-RP-DE	344.43	13.18%	1.82	2.24%	0.64	0.84%
DE-DE-DE	1740.68	66.60%	78.46	96.53%	74.38	98.51%
Total	**2613.48**	**91.79%**	**81.28**	**2.85%**	**75.50**	**2.65%**
Total Run-time	**2847.17**	(Overhead 76.90 sec 2.70 %)				

The breakdown of the number of rays and the run time is given graphically in figures A.7(a) and A.7(b) respectively. Of the tertiary rays 99 % are geometrically invalid, less than 1 % are geometrically valid and less than 1 % are not shadowed. Path finding time takes 94 %, the shadow tests 3 % and the EM calculations 3 % of the run time.

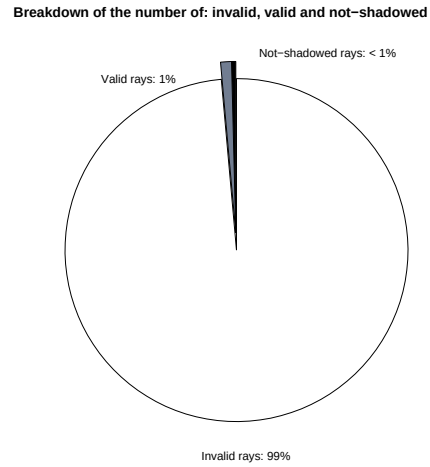


(a) Breakdown of the number of: invalid, valid and not-shadowed rays.

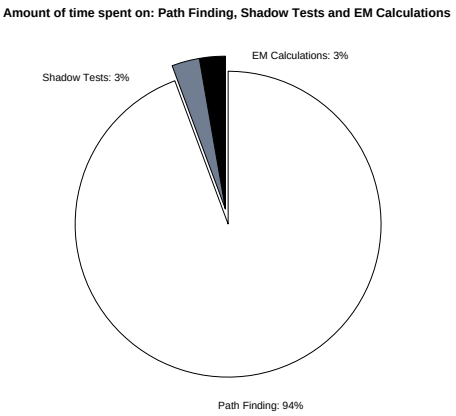


(b) Breakdown of the time spent on: path-finding, shadow tests and EM calculations.

Figure A.6: Breakdown of counting and timing results for combined ray-types.



(a) Breakdown of the number of: invalid, valid and not-shadowed rays.



(b) Breakdown of the time spent on: path-finding, shadow tests and EM calculations.

Figure A.7: Breakdown of number of rays and run time for tertiary ray-types.

A.3.3 Discussion of the Empirical Results

Simple Rays

From the results presented in section A.3.2 the following observations are made when a model is run using only simple ray.

- The majority of rays ($\approx$ 85 %) are geometrically invalid (exceptions are direct and DEC rays).
- Approximately 35 % of the run time is spent on path-finding.
- Of the geometrically valid rays, approximately 60 % are shadowed.
- Just over 50 % of the run time is spent on the shadow tests.
- EM calculations take up 12 % of the run time which is less than path finding and the shadow tests.

Using only simple rays, the shadow tests take up the majority of the run-time. Path finding however still takes up a large portion of the processing time.

Combined Rays

The results for the combined rays in section A.3.2, show that the path finding becomes more important as the complexity of the rays increases. The following observations are made.

- The majority of rays ($\approx$ 98 %) are geometrically invalid (exceptions are direct rays, DEC and DL-DL ray-types).
- The majority of processing time (approximately 92 %) is spent on path-finding. This is approximately 3 times the path finding percentage for simple rays.
- Of the remaining valid rays, approximately 80 % are shadowed. Examining the number of rays that are shadowed, shows that more second order rays are shadowed than simple rays.
- The run time spent on the shadow tests (approximately 7 %) is negligible compared to the path finding time.
- Time spent on EM calculations is less than one percent of total runtime.

When using more complex rays, path finding time dominates and the shadow tests time becomes negligible.

Tertiary Rays

The results for tertiary rays are similar to the combined rays.

- The majority of rays are geometrically invalid.
- 92 % of processing time is spent on path finding.
- 62 % of the geometrically valid rays are shadowed.
- The shadow test time is negligible when compared with the path finding time.
- The time for EM calculations is also negligible when compared with the path finding time.

Discussion

The tests show that to optimise SuperNEC-UTD both the path finding algorithm and the shadow tests need to be optimised. In all tests, these 2 components take up over 85 % of the processing time. Until path finding and the shadow tests are optimised, the EM calculations can be ignored. Using simple rays both path finding and the shadow tests take up a significant portion of the processing time. Using more complex rays (combined and tertiary ray classes) the path finding time dominates and the shadow tests use a negligible portion of the processing time. These results indicate that any optimisations should first aim to reduce the path finding time as this would benefit both simple and complex ray types. However it is the second and higher order rays that cause the large path finding times. A second optimisation should focus on reducing the shadow test time.

A.4 Individual Analysis of the Ray-Tracing Code

The previous section showed that path finding and the shadow tests have excessive run times. Using the results from section A.3 the 2 ray tracing sections of SuperNEC-UTD are analysed and discussed individually. This will give an understanding of why these parts are slow.

A.4.1 Path Finding

Path finding searches for geometrically valid paths by exhaustively interpolating ray paths between all components of the model. The large number of combinations of possible rays causes path finding to spend a large amount of time determining invalid paths. The results in section A.3 indicate that the run time for path finding for higher order rays is much higher than for simple rays and that many of the paths found are geometrically invalid. In this section, bottlenecks in path finding are identified. The search space is explained in more detail and symmetry in ray paths is discussed.

Identification of Bottlenecks in Path Finding

Inspection of the source code has revealed that the excessive run-time of path-finding is due to a combination of factors.

- Large search space, ie. number of possible path combinations, increases rapidly with model size (see section A.2.2 for a description of path finding).
- When the impedance matrix is filled using UTD, the symmetry of the ray paths is ignored.
- Higher order rays use iterative methods. These rays are: DE-DE, RC-RC, RC-DC, DC-RC, DC-DC, RC-DE, DE-RC, DC-DE, DE-DC.
- Slow execution due to the mathematics involved which requires many floating point operations.

The slow execution times of the iterative methods and slow mathematical functions involved in path finding are exacerbated by the large search space. A large combination of ray paths means that the iterative methods and the slow mathematical functions are called repeatedly. When the symmetry of different ray paths is not exploited the same ray paths are often calculated twice.

Dominate Ray Types DE-DE, DE-RC, RC-DE and DE-DE-DE rays dominate path finding (see the timing results for combined and tertiary rays in tables A.9 and A.11). Analysis of these rays has identified 2 bottlenecks.

1. `Edge::ComputeIlluminatedAngle` which determines for an edge, the direction of the incident wave and the o-face and n-face of the diffraction wedge (see [6, chp. 4]). These ray-types, use an iterative method to find the points of reflection and diffraction.
2. `Cylinder::ReflectionPointNF` which determines the point of reflection on a cylinder given the source and observer points.

A closer inspection of the source code has shown that these 2 functions must be called in each loop of the iterative methods. The functions cannot be moved outside of the iterations.

Search Space in Path Finding

The reason for the large difference in time for path finding for simple rays and for higher order rays is that the number of combinations of paths for simple rays is much smaller than for higher order rays. This section explains the causes of the large number of paths for higher order rays and gives results from simulations which investigated the search space.

Relationship between the search space and the model size. For a fixed source and a fixed observer locations, there is only 1 possible direct ray path. Now consider for instance double reflection between 2 plates (RP-RP). The maximum number of paths is given by the number of permutations, ie."the number of ways of obtaining an ordered subset of elements from a set of elements" [8]

Table A.12: Example search space sizes for a model consisting of a single segment and 49 plates where each plate has 3 edges.

Direct Ray	RP	DE	RP-RP	RP-DE	DE-RP	DE-DE	DE-DE-DE
1	49	147	2352	7056	7056	21462	3133452

$$ {}_{n}P_{k} \;=\; \frac{n!}{(n-k)!} \tag{A.1} $$

Where ${}_{n}P_{k}$ are the number of permutations, n the number of elements in the set and k the number of elements in the subset.

Given a fixed source and a fixed observer location, for 49 plates (ie. $n = 49$, $k = 2$[1]), there are at most $49!/(49-2)! = 2352$ double plate reflection paths between the source and observer. SuperNEC-UTD will search all of these combinations, many of which are not geometrically possible (see section A.2.2).

For RP-DE and DE-RP rays the number of paths is expressed by

$$ \text{Number of paths} \;=\; p \times (e \times (p-1)) $$

Where p is the number of plates and e is the number of edges per plate[2]. For a model with 49 plates and 3 edges per plate, there are up to 7056 possible RP-DE and DE-RP ray paths.

For double edge diffraction rays (DE-DE) the search space size is calculated as follows. Given 49 plates and assuming each plate has 3 edges, there are a total of $49 \times 3 = 147$ edges. Using equation A.1 this gives, $147!/(147-2)! = 21462$ permutations. SuperNEC-UTD will search all 21462 double edge diffraction paths for valid paths.

Triple diffraction rays consist of 3 paths. The first path is from a fixed source to an edge. There are

$$ p \times e $$

number of paths from the source to all plate edges. p is the number of plates and e the number of edges per plate. From the first edge to the next edge there are

$$ p \times e - 1 $$

number of paths. The -1 indicates that the ray cannot diffract off the same edge consecutively. The number of paths from the 2nd to 3rd edge is also given by the previous equation. If the observer is fixed, the path from the 3rd edge to the observer does not increase the search space. The total number of paths for triple diffraction rays is then given by

$$ p \times e \times [p \times e - 1]^2 $$

For a model with 49 plates and 3 edges per plate, there are 3133452 (approximately 3 million) DE-DE-DE rays.

Table A.12 summarises these results. For each ray type the maximum number of paths from a fixed source and observer are given. Path finding takes longer for more complex rays because the search space increases in an exponential manner.

Relationship between search space and the number of segments. When determining the coupling between the segments, the search space for each ray-type is proportional to n^2 (where n is the number of segments). This relationship is explained as follows.

Each ray has a source segment and an observer segment. In the case of self coupling of a segment the source and observer are the same. The number of self coupling rays for each ray-type is proportional to n. Mutual coupling involves a source which is different from the observer. Each source radiates to $n-1$ segments. There are n segments and therefore the number of mutual coupling rays for each ray type is proportional to $n \times (n-1)$. The sum of

[1] $k = 2$ because for a double reflection ray 2 combinations of plates are required.

[2] In general, the number of edges per plate is not constant. The formula only gives an indication of the size of the search space for RP-DE and DE-RP rays.

Table A.13: Test models used for combined path finding tests.

(a) Test models with increasing model size.

	Models (increasing size)							
	1	2	3	4	5	6	7	8
Frequency	300 MHz							
Segments	7 segments							
Plates	7	14	21	28	35	42	49	56
Cylinders	1	1	2	2	3	3	4	4

(b) Test models with increasing number of segments.

	Models (increasing number of segments)							
	1	2	3	4	5	6	7	8
Frequency	300 MHz							
Segments	7	12	17	22	27	32	37	42
Plates	34							
Cylinders	3							

Table A.14: Path finding time and number of rays for 8 test models. Time is given in seconds.

(a) Increasing model size.

Model	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Time	47.61	120.71	350.56	370.75	990.08	1503.60	1921.21	3051.58
Rays	2325400	8720250	13405700	28432250	47570600	73692850	98660100	136601850

(b) Increasing number of segments.

Model	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Time	965.57	1901.68	2676.81	3500.35	4391.50	5299.09	6272.93	7303.56
Rays	46696650	80779140	115468080	150763470	186665310	223173600	260288340	298009530

the self and mutual coupling rays equals n^2 which means that the number of rays for each ray-type is proportional to n^2.

For the example given in table A.12, the search space of each ray-type must be multiplied by n^2 to get the correct number of rays (excluding direct rays) for the coupling between segments. The example in table A.12 assumed a single segment. When multiple segments are present the coupling combinations between segments increases the number of rays. In general the number of segments in a UTD model is low and the n^2 relationship between the number of segments and search space can be neglected.

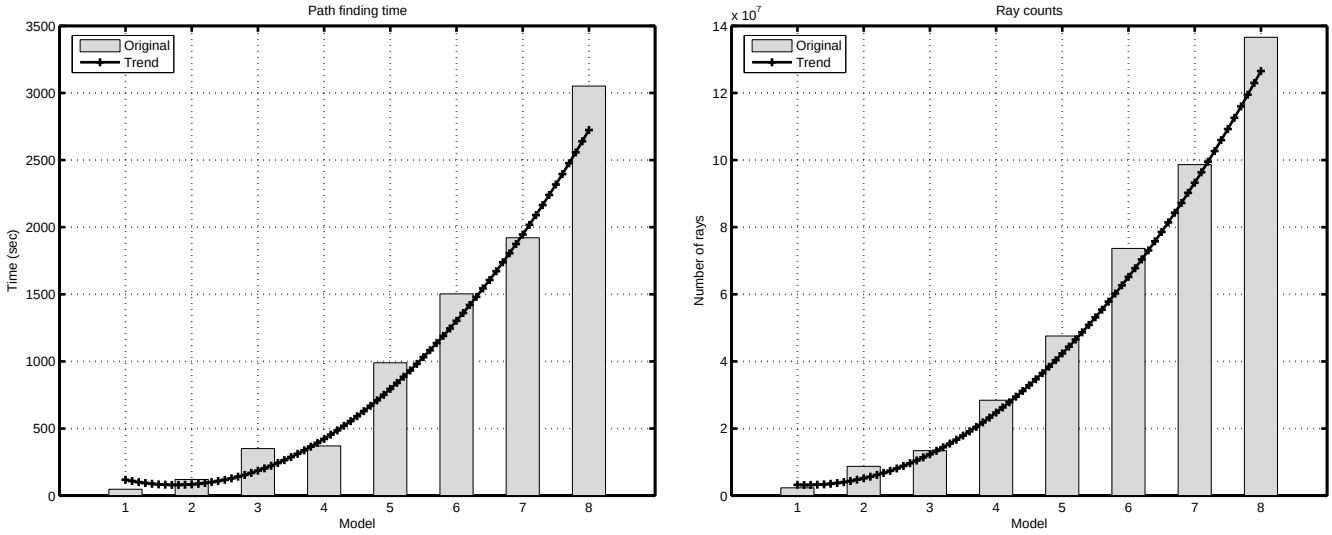
When calculating the radiation pattern the relationship between the number of segments and the search space is linear. The number of far field points (f) is assumed fixed. If there is one segment, there are $1 \times$ f direct rays. There are $1 \times 49 \times$ f RP rays for the example in table A.12. Similarly for the other ray-types in the table. If there are n segments the search space for each ray-type increases linearly, ie. there will be $n \times$ f direct rays. For RP rays in the example there are n $\times$ 49 $\times$ f rays. The search space when calculating the far field radiation patterns is proportional to n.

Empirical results. From the above discussion the search space increases exponentially with model size and and linearly with the number of segments. This was investigated and the results are presented next. The test models used to examine the nature of the path finding time are given in table A.13.

Two sets of models are used. In the first set (table A.13(a)) the number of plates and cylinders is increased while the number of segments is constant. In the second set (table A.13(b)) the number of segments is increased while the number of components is constant.

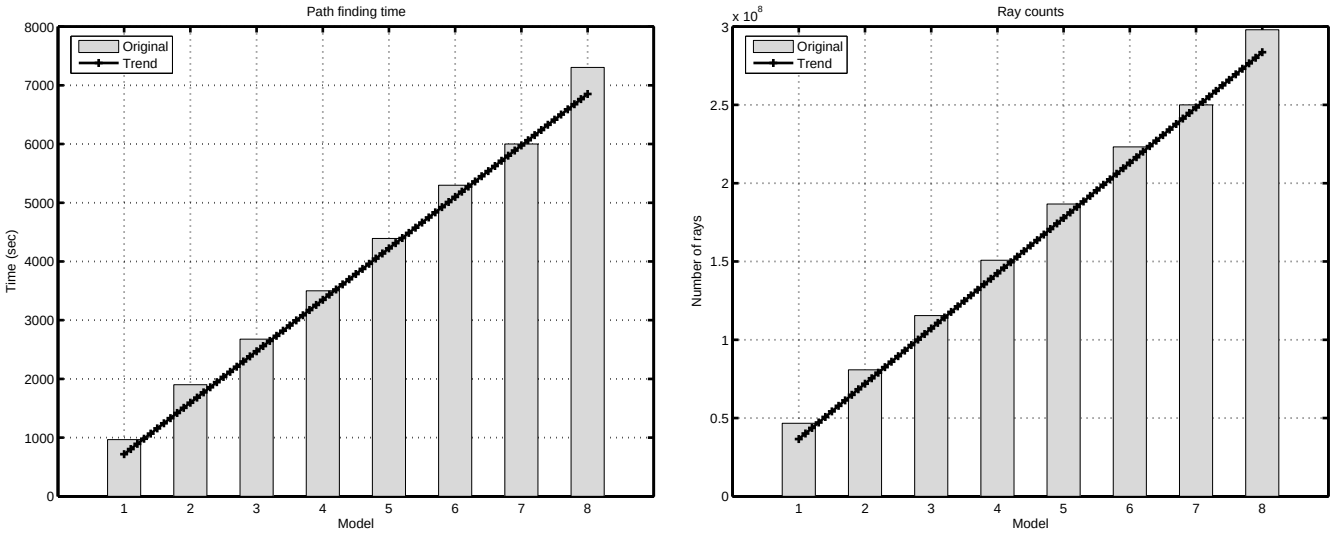
Table A.14(a) lists the path finding time and the number of rays for the 8 models with varying number of components. The path finding time and the number of rays are plotted in figure A.8. The results for the test models with varying number of segments are given in table A.14(b). In figure A.9 the path finding time and the number of rays are plotted.

The trend lines in figure A.8(a) and A.8(b) indicate that as the model size increases the



(a) Path finding time. (b) Number of rays.

Figure A.8: Path finding time and number of rays for 8 test models for increasing number of segments.



(a) Path finding time. (b) Number of rays.

Figure A.9: Path finding time and number of rays for 8 test models for increasing number of segments.

path finding time and the search space increase exponentially. In figure A.9(a) and A.9(b) the trend lines indicate that the size of the search space is proportional to the number of segments. This corresponds with the discussions in the previous paragraphs. The excessive run time of path finding is a result of the exponential growth of the search space as the number of components increases.

Symmetric Ray Paths

When filling the impedance matrix in MoM the symmetry associated with rays is ignored.

A direct ray from source A to observer B follows the same path as a ray from observer B to source A. This means only one path needs to be found. Similar for more complex rays, eg. a double edge diffraction from source A over edges E_1 and E_2 to observer B follows the same path as the reverse ray from the observer to the source, ie. the points of diffraction do not change. This symmetry cannot be directly used with rays that travel over different component types, eg. a RP-DE ray from source A, reflecting off plate P_1 and diffracting off edge E_1 to observer B, equals a DE-RP ray from observer B, diffracting off edge E_1, reflecting off plate P_1 and arriving at source A. In other words every RP-DE ray is the inverse of a DE-RP ray.

A.4.2 Shadow Tests

The shadow tests determine if a ray intersects a component. As explained in section A.2.2, the shadow tests use a brute force algorithm which does not scale with model size. In section A.3, results showed that for simple rays, the shadow tests have a high run time. When the number of rays increases by increasing either the number of components in a model or the number of segments, the time for the shadow tests increases exponentially and does not scale. Scalability of the shadow tests and the relationship between the shadow test time and the complexity of the rays is discussed in this section.

Scalability

Two sets of experiments are run to investigate the relation between the shadow test time and 1) the model size and 2) the number of segments. 8 models are used which consist of only plates. Only direct, single diffraction and single reflection rays are used. The reasons for using simple rays and only plates during the tests is explained next.

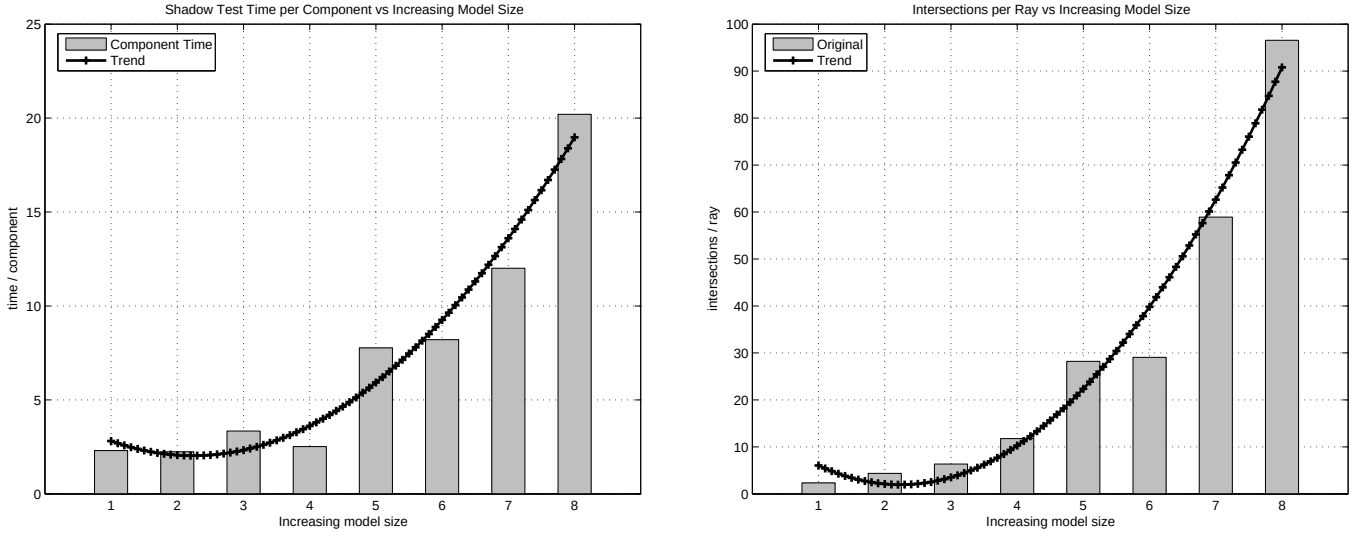
The ray types used during the experiments is irrelevant. A ray is passed to the shadow algorithm which determines if this ray is blocked. The shadow algorithm is not aware of the type of ray. Therefore only simple rays (direct, single diffractions and single reflections) are used in the tests. The models consist only of plates. The shadow algorithm tests a ray against each component in the model. Whether the component is a cylinder or a plate is irrelevant. The time for a single intersection test for a plate does not equal the time of an intersection for a cylinder but the relationship between model size and shadow time will be the same whether cylinders, plates or a combination of both are used. Plates are used to keep the tests simple.

Table A.15: Scalability analysis for shadow algorithm with increasing model size.

Rays	direct, DE, RP							
Frequency	1000 MHz							
Radiation Patterns	3 dimensional radiation pattern with 1 degree increments.							
Plates	3	6	14	25	35	49	98	147
Shadow Time (sec)	244.27	420.42	2000.42	1750.91	8088.61	12745.85	38023.31	92362.74
Time/Plates (sec/plate)	81.42	70.07	142.89	70.03	231.10	260.12	387.99	628.32
Intersections/Ray	2.33	4.36	6.36	11.77	28.22	29.06	58.92	96.57

Model Size In the first experiment the relationship between model size and shadow time is investigated. The results are recorded in table A.15. The simulation frequency is 1000 MHz and the number of components in the models range from 3 to 147 plates. A 3 dimensional

radiation pattern (the elevation and azimuth angles are both incremented by 1°) is calculated for each model. The shadow time, the average shadow time per plate (shadow time divided by the number of plates) and the average number of intersections per ray (total number of intersections divided by the total number of rays) are recorded.



(a) Shadow time per component.

(b) Average intersections per ray.

Figure A.10: Scalability analysis for shadow algorithm with increasing model size.

The time per plate and the intersections per ray are plotted in figures A.10(a) and A.10(b) respectively. The graphs both increase quadratically as indicated by the trend lines. The shadow test time per plate and the number of intersections per ray are proportional to c^2, where c is the number of components in a model.

The reason for the exponential growth of the shadow test time with model size, is that every time a component is added to the model, not only does the algorithm check each ray with new the component for an intersection but also the number of possible paths increases which leads to an increase in the number of rays.

Consider a model with c components. The algorithm must do in the worst case c intersection tests for each ray. This is the case when the ray intersects no components or the ray intersects the last component. Increasing the number of components in a model, increases the number of possible ray paths between a fixed source and observer. In a model with only a single plate, there can only be a single reflection from a fixed source to and fixed observer. This means, there is only one ray and one component and therefore only one intersection test. With 2 plates, there could be 2 reflection paths from the fixed source to the fixed observer points. For each ray there are at most 2 intersection tests (one for each plate). Therefore, a total of 4 intersection tests. In other words the number of ray paths increase with model size which means the number of shadow tests increases quadratically with model size.

Table A.16: Scalability analysis for shadow algorithm with increasing number of segments.

Rays	direct, DE, RP							
Number of plates	98							
Radiation Patterns	3 dimensional radiation pattern with 1 degree increments.							
Frequency (MHz)	250	500	750	1000	1250	1500	1750	2000
Segments	10	18	26	34	42	54	62	70
Shadow Time (sec)	340.81	622.32	900.94	1179.20	1188.46	1879.24	2159.41	2429.27
Intersections / Ray	58.97	58.96	58.93	58.92	58.91	58.91	58.78	58.66

Number of segments In the second experiment the relationship between the shadow time and the number of segments is investigated. The results are recorded in table A.16. The number of components is kept constant and the number of segments increases from 10 to 70. A 3 dimensional radiation pattern is calculated. During the calculations the shadow time,

total number of intersection tests and the total number of rays is recorded. In table A.16 the shadow time and the average number of intersections per ray (intersections/ray) are given.

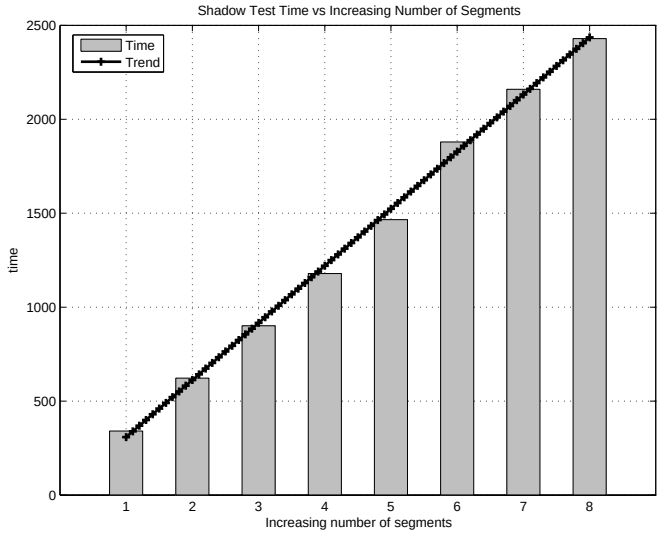


Figure A.11: Shadow time vs increasing number of segments.

The shadow time is plotted in figure A.11.

Increasing the number of segments causes a proportional increase in the shadow time. Adding another segment to a model means more rays will be traced from the new segment. However adding a segment does not increase the search space. The number of paths from the new segment to the observer points remains approximately constant for all segments and therefore each segment launches approximately the same number of rays. If one segment launches k rays, then two segments launch 2k rays, etc. The relationship between number of segments and model size is linear. The average number of intersection tests per ray, is approximately constant for all models which means that all the rays are tested against an equal number of components.

The above 2 experiments show that the shadow time increases exponentially with model size and linearly with number of segments.

Shadow Tests and Ray Complexity

The majority of geometrically valid rays are shadowed (between 60 % and 80 % of the geometrically valid rays are shadowed during the tests). The percentage of not shadowed rays decreases for higher order rays. Simple rays (eg. direct rays, single reflections or single diffractions) are less likely to be shadowed. Higher order rays are more likely to fail the shadow tests, because they consist of more paths. A direct ray only consists of a single path. A triple plate reflection consists of 4 paths. The probability that a triple RP ray is shadowed is thus 4 times higher than for a direct ray. Similar arguments hold for other higher order rays.

A.5 Conclusion

SuperNEC-UTD consists of 3 main sections: path finding, shadow tests and electromagnetic calculations. An overview is given on how SuperNEC-UTD works. Results of tests conducted are presented which highlight the bottlenecks. The major bottlenecks are path finding and the shadow tests. Both are implemented using a brute force algorithm and their run time grows exponentially with model size.

Tests show that for simple rays the shadow tests take up a large percentage of the total run time. A brute force shadow algorithm determines if a ray intersects a component by

arbitrarily comparing that ray against every component until an intersection is found or until all components have been exhaustively examined

Path finding time dominates the overall run time when higher order rays are used. Higher order rays consist of multiple coupling mechanisms. For a given pair of source and observer points, the path finding algorithms examines all possible geometric paths. For higher order rays there are many paths which leads to high run times.

The simulations show that no single optimisation technique will optimise SuperNEC-UTD. Only a set of optimisation techniques used conjointly will reduce the excessive run time of the program. Only focusing on the shadow tests, will ignore the excessive run times of path finding for higher order rays. Focusing uniquely on optimising path finding will overlook the importance of shadow tests when using simple rays.

To improve the performance of SuperNEC-UTD, first the path finding algorithm must be optimised. Path finding can be optimised by reducing or limiting the search space that needs to be examined for a source and observer pair. Secondly the shadow tests must be optimised. An optimisation technique for the shadow tests should ideally reduce the number of intersection tests required per ray and be independent of model size.

References

[1] SuperNEC. Poynting Antennas and Electromagnetics. Last accessed 15 June 2006. [Online]. Available: http://www.supernec.com

[2] A. Fourie and D. Nitch, "SuperNEC: Antenna and indoor propagation simulation program," *IEEE Antennas and Propagation Magazine*, vol. 42, no. 3, June 2000.

[3] R. G. Kouyoumjian and P. H. Pathak, "A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface," *Proceedings of the IEEE*, vol. 62, no. 1, Nov. 1974.

[4] J. Keller, "Geometrical theory of diffraction," *Journal of the Optical Society of America*, vol. 52, no. 2, pp. 116–130, 1962.

[5] R. F. Harrington, *Field Computation By Moment Methods.* New York: Macmillan, 1968.

[6] D. A. McNamara, C. W. I. Pistorius, and J. A. G. Malherbe, *Introduction to the Uniform Theory of Diffraction.* Artech House, 1990.

[7] Intel pentium 4 processor. Intel Corporation. Last accessed 3 April 2006. [Online]. Available: http://www.intel.com/products/processor/pentium4/

[8] E. W. Weisstein. Permutation. MathWorld. Last accessed 27 May 2006. [Online]. Available: http://mathworld.wolfram.com/Permutation.html

Appendix B

Path Finding Optimisations for SuperNEC-UTD

Abstract

In SuperNEC-UTD, geometric optimisations based on back-face culling, image theory and the theory of diffraction are used to reduce the search space for rays that interact with plates. Originally, SuperNEC-UTD uses an exhaustive linear search to examine all possible ray path combinations and to find geometrically valid paths. Back face culling (BFC) is used to determine for each segment, which plates of a polyhedron are visible. No reflection and diffraction rays need to be traced from a segment to a plate that is not visible. BFC optimisations can only be used for first order plate reflections or diffractions and the technique is only valid for polyhedrons. Using image theory and the theory of diffraction the region and components seen by a reflection or diffraction are determined in a preprocessing step. During path finding, rays need only be traced to regions (called reflection and diffraction zones) and components illuminated by a ray. The reflection zones (RZ) can only be used with consecutive reflections from a segment. Diffraction zones (DZ) can only be used with wedges. BFC and DZ introduce errors, because corner diffracted rays are discarded. All the optimisations involve plates. Path finding optimisations for cylinders have not been implemented but the techniques can determine when a cylinder is illuminated by a plate reflection or a plate diffraction. When used collectively on generic models the contribution of each optimisation to the overall speed-up is as follows: on average the original run time is reduced by 17 % using BFC, 4 % using RZ and 27 % using DZ. The mean absolute error for BFC is approximately 1.6×10^{-2} dB and for DZ about 1.79×10^{-2} dB. Overall the original run time is reduced by 46 % and the mean absolute error is about 5.55×10^{-2} dB. The errors are very small and the optimised radiation patterns follow the original graphs very closely.

B.1 Introduction

SuperNEC [1, 2] is a numerical electromagnetic software simulator that includes an implementation of the uniform theory of diffraction (UTD) [3, 4]. The UTD code uses ray-tracing [5, 6] to determine the paths of high frequency electromagnetic waves. Path finding in SuperNEC-UTD is the process of finding a geometric path between a source and an observer point in a model made up of plates and cylinders. The path can include zero or more reflections and zero or more diffractions.

Path finding in a model that consists of a large number (> 50) of components, is slow. The number of possible paths a ray could follow from a source to an observer grows exponentially with the number of components in a model. The original path finding algorithm exhaustively examines all possible ray-paths which means that the algorithm does not scale with model size. The problems with path finding are discussed in detail in [7].

This report presents 3 solutions to the path finding problem. They are 1) back-face culling [8] using polyhedron identification, 2) reflection zones using image theory [9] and 3) diffraction zones using the theory of diffraction [3, 10]. All 3 techniques reduce the number of possible paths that the path finding algorithm has to examine. Each technique is used to optimise different ray-types.

An overview of the optimisations is given in section B.2. Sections B.3 to B.5 present the principles behind back face culling and reflection and diffraction zones. In section B.6 the combined performance of the techniques is discussed. Section B.7 summarises the 3 optimisation techniques.

B.2 Solutions

B.2.1 Optimisation Principles

All solutions to the path finding problem have 2 common characteristics.

1. space / time trade-offs
2. search space reduction

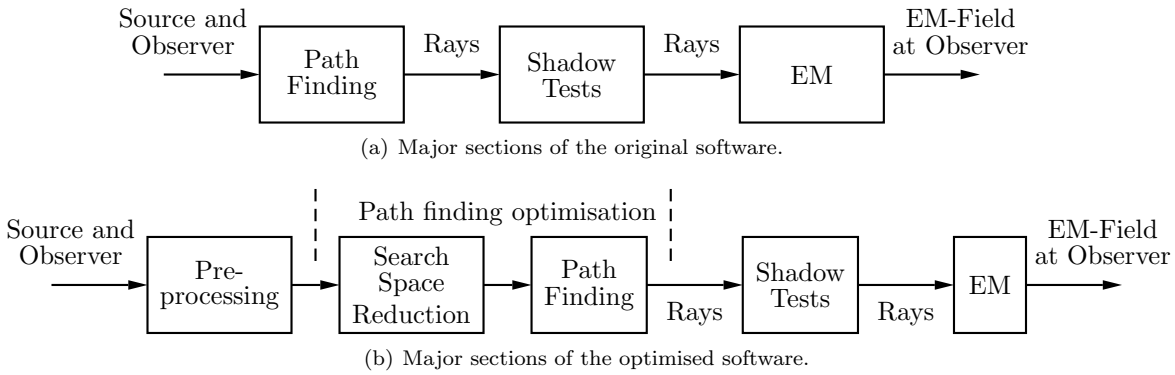
In a pre-processing stage, the geometry of the SuperNEC model is analysed and an information structure is built. The information structure is stored in memory for the duration of the simulation. During path finding the information is used to reduce the search space. Given a source point, an observer point and one or more components which lie in between those 2 points, the information structure is used to determine in advance whether a ray between the source and the observer will be geometrically valid. This means fewer rays are traced. Memory is used to store the information structure. Later, during path finding less time is spent examining ray paths which are geometrically invalid: the *search space* is reduced. Accessing the information structure is overall faster than examining all possible ray paths: *space is traded for time.*

The solutions presented in this document solve the path finding problem globally. Various ray-types are used in SuperNEC-UTD. It is possible to focus on the individual ray-types and their idiosyncrasies, eg. optimising the iterative methods or limiting the number of times complex mathematical functions are called [7]. The solutions presented in this document exploit features which are common to multiple ray-types.

Block diagram of SuperNEC-UTD

Figure B.1 shows the structure of SuperNEC-UTD. In figure B.1(a) the block diagram of the original software is shown. The software consists of 3 parts: path finding, shadow tests and electromagnetic (EM) calculations which is discussed in [7].

In the optimised software, shown in figure B.1(b), a pre-processing stage is added which initialises the data structures for the 3 optimisation. Before path finding, the data structures



(a) Major sections of the original software.

(b) Major sections of the optimised software.

Figure B.1: Major sections of SuperNEC-UTD. The original software consists of 3 sections: path finding, shadow tests and electromagnetic (EM) calculations. In the new software a preprocessing stage and path finding optimisations have been added.

are used to reduce the search space. During path finding only geometrically valid rays are traced.

B.2.2 Outline of the Optimisation Techniques

The optimisations (back face culling, reflection zones and diffraction zones) are presented in the next 3 sections. Each section is structured identically. First the general operation of the optimisation without reference to SuperNEC is presented. Then follows a section on how the technique is employed in SuperNEC.

B.3 Back Face Culling

In this section back face culling is explained. Back face culling eliminates those plates of a polyhedron which are not visible from a segment. Using back face culling, rays are only traced to plates of a polyhedron which are visible.

B.3.1 Principle of Operation

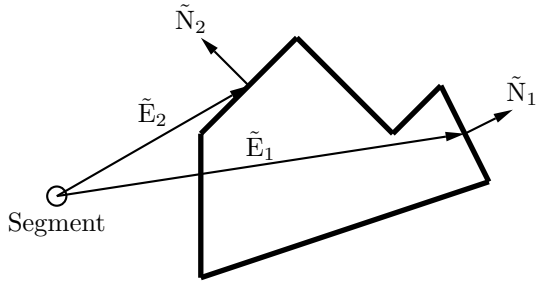


Figure B.2: Back face culling illustrated for the 2 dimensional case. Adapted from [8]

Figure B.2 illustrates the principle used to cull invisible faces from a polyhedron. The concept illustrated for the 2 dimensional case in figure B.2 can be used directly with polyhedrons.

In a closed structure (ie. a polyhedron), those polygons which are not visible from a segment are detected using the dot product of the normal of the polygon and the vector from the segment to a point on the polygon. In figure B.2 $\vec{N}_1$ and $\vec{N}_2$ are the normals of 2 sides of the structure. A vector is drawn from the segment to each normal. If the angle between the normal and the vector from the segment is $\geq -90°$ and $\leq 90°$ the side is visible. This is the case for the side with normal $\vec{N}_2$. Otherwise the side is not visible, as is the case with

normal $\vec{N}_1$. The angle between the 2 vectors is given by the dot product, thus

$$\text{A side is visible if } \tilde{E}_x \cdot \tilde{N}_y \ \leq \ 0$$

The direction of the normals must be the same for all sides of the polyhedron. In the figure, the normals point towards the outside. When $\vec{E}_x \cdot \vec{N}_y \ = 0$ then the side is viewed edge-on. In this report, the side is assumed visible if viewed edge-on.

B.3.2 Optimisation

Using back-face culling, the plates visible from a segment are identified. Rays need only be traced from a segment to a plate, if the plate is visible. This improves the path finding for rays that either reflect or diffract at the first plate they intersect, eg. DE-x or RP-x rays. Where x can be any other higher order coupling mechanism, eg. cylinder reflection, edge diffraction etc. Of course back face culling also works for single DE and RP rays.

Back face culling trades time for memory by storing a list of visible plates for each segment. When tracing a ray only those plates are considered which are visible from the plate. The search space for path finding is reduced because fewer plates need to be examined which translates into faster path finding.

Relevant Geometric Structures Back face culling can only be used with polyhedrons. The technique cannot be used with open structures, single plates and cylinders.

Relevant Rays Rays which can be optimised using back face culling are listed in table B.1. The first interaction of these rays is either an edge diffraction or a plate reflection.

Table B.1: Relevant rays for back face culling.

Primary and Secondary Rays		**Tertiary Rays**	
Reflection	**Diffraction**	**Reflection**	**Diffraction**
RP, RP-RP, RP-DE, RP-RC, RP-DC,	DE, DE-RP, DE-DE, DE-RC, DE-DC	RP-RP-RP	DE-RP-RP, DE-RP-DE, DE-DE-RP, DE-DE-DE

B.3.3 Back Face Culling in SuperNEC-UTD

Back face culling is implemented in 4 steps.

1. Identify common edges.
2. Identify polyhedrons in the model.
3. Determine the visible plates for each segment.
4. Ray trace only to the visible plates

Identify common edges. All the plates in a model are examined to determine if the edge of one plate is connected to another plate. If the edge is connected to another plate, then both edges are flagged as connected. An edge is *aware* of the edge and plate that is is connected to. The code to identify common edges is already implemented in the original SuperNEC.

Identifying the polyhedrons in a model is done using a recursive algorithm. If a plate is part of a polyhedron, then all the edges of the plate must be connected to other plates. All those plates in turn must be connected to other plates. A recursive algorithm is used to *traverse* the polyhedron by starting at some arbitrary plate (of the polyhedron) and moving from each edge (of the plate) to the next plate. By ensuring that each edge of a plate is connected to another plate and that the connected plates are also connected to subsequent plates, closed structures are identified.

Determining visible plates. For each plate that is part of a polyhedron the test described in section B.3.1 determines whether that plate is visible from a segment. This is done for each segment. The result is a list of visible plates associated with each segment.

In the optimisation, the dot product is taken between the vector from the segment to a corner of the plate and the normal vector of the plate. A plate is declared not visible from the segment, if the dot products for all corners of a plate fail the visibility test.

Ray tracing. When tracing a ray from a particular segment, only plates of a polyhedron which are visible from that segment are considered. Single plates, or plates which are part of an open structure, are assumed visible.

B.3.4 Errors for Back Face Culling

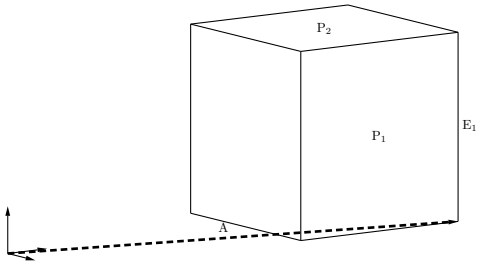


Figure B.3: The cause of the errors in back face culling.

Diffraction off the corner of an non-visible edge Only in ray-types which start with a diffraction do errors occur. Ray A in figure B.3, illustrates the case when a ray diffracts off the corner of an edge. The ray originates at a segment (shown as a small coordinate axes). The segment is situated below the bottom of the cube and to the left of the cube. The diffraction edge is labelled E1. Ray A cannot see the edge but the corner is visible. In the optimised software the edge E1 is not visible because plate P1 is not visible. So in the optimised software the ray cannot diffract off the corner. This causes the errors in back face culling. However, in the original program, the brute force nature of path finding, means that the corner diffracted ray is found.

As frequency increase, corner diffracted fields decay faster than edge diffracted fields [10]. Corner diffracted rays show up as discontinuities in the radiation patterns [10]. However, the overall radiation pattern still provides a useful and accurate solution for engineering purposes. Considering the speed-up achieved using back face culling, the errors are an acceptable compromise.

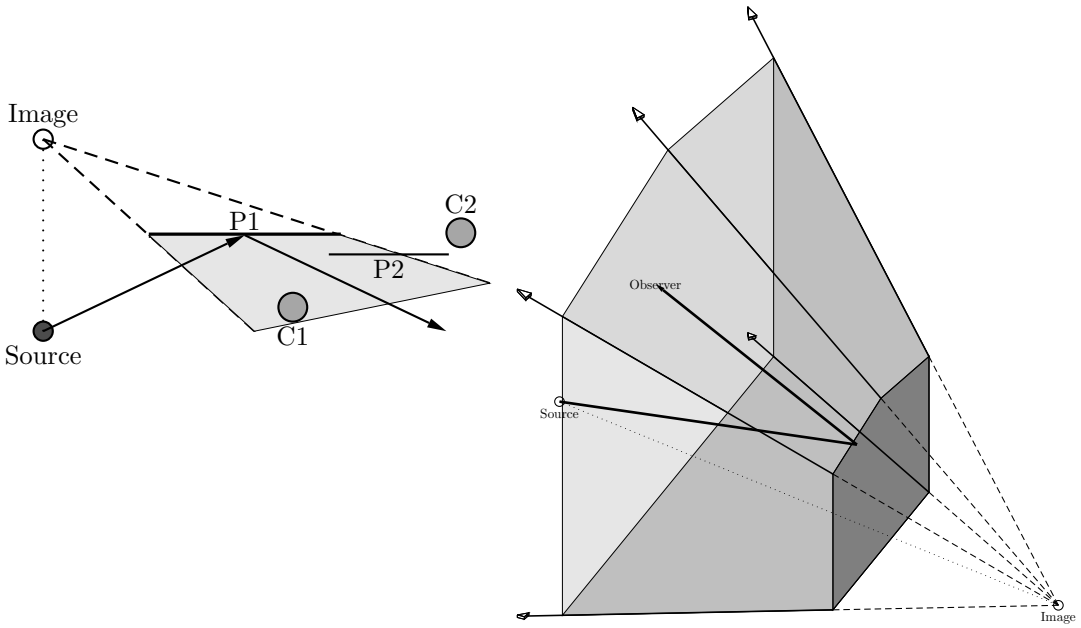
B.4 Reflection Zones

In this section path finding optimisations for reflections rays are explained. The optimisations in this section are taken from [9]. Reflection zones are used to determine in advance the regions which can be reached by a reflecting ray. Using the reflection zones only geometrically valid reflection rays are traced.

B.4.1 Principle of Operation

Reflection zones are based on image theory. Using image theory the spatial region which can be reached by a ray reflecting in a plate is constructed. The spatial region is called a reflection zone. The concept of the reflection zone is demonstrated for the 2 dimensional case in figure B.4(a).

In figure B.4(a), all reflections from a point source in a plate seem to originate from the image of the source in the plate. The reflecting plate and the image source together define



(a) Two dimensional reflection zone. Adapted from [9].

(b) Three dimensional reflection zone. Adapted from [9].

Figure B.4: Reflection Zones (2D and 3D)

the spatial region which can be reached by a ray originating at the source and reflecting off the plate. In figure B.4(a) a ray from the source reflects on plate P1. Rays from the source can only be reflected into the shaded area.

A reflection zone is constructed by determining the image of the source. The image lies on the line perpendicular to the plate and which connects the source and image. The image is the same distance from the plate as the source is from the plate. The shaded area called the reflection zone is determined by extending imaginary lines from the image to the vertices of the plate. Valid reflections can only occur in the reflection zone defined by the imaginary lines and the plate. The zone does not include the image point.

Two dimensional reflection zones can be directly extended to the third dimension as shown in B.4(b). The 3 dimensional reflection zone is the shaded semi-infinite volume. A ray reflecting in the plate can only reach the shaded volume. Plane equations are used to define this volume[1]. Higher order reflection zones can be constructed in a recursive manner by determining images of images.

B.4.2 Optimisation

The reflection zone is helpful in 2 ways. Firstly, an observation point can only be reached by a reflection, if the observation point is in the reflection zone. Before tracing a ray from a source, reflected by a plate to an observer, a simple test determines if the observer is in the reflection zone. Impossible reflections are therefore quickly eliminated.

Secondly, by determining the reflection zone, the components in that region can be identified. In figure B.4(a) only plate P2 and cylinder C1 are in the reflection zone. Cylinder C2 is outside. This is useful when after the first reflection, the ray reflects or diffracts off a second component. For subsequent reflections or diffractions the ray is traced only to components which are within the spatial region. Components outside the region are ignored.

Higher order reflection zones offer similar optimisations. By determining in advance the valid reflection zones, the search space for path finding for reflection rays is reduced.

[1]The normals of the planes must be defined to either all point into or outside the reflection zone.

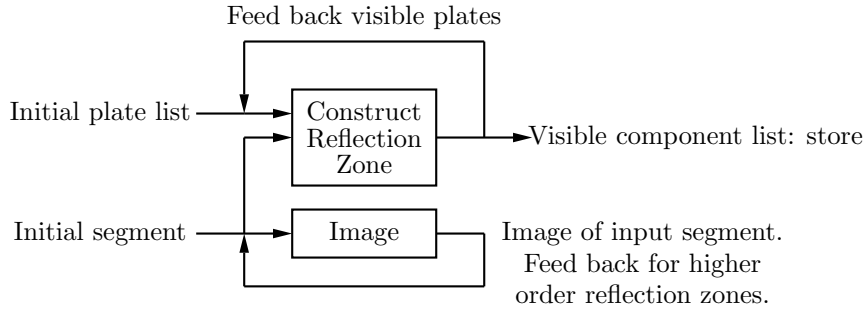
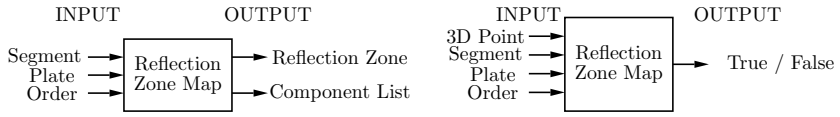


Figure B.5: Block Diagram of the recursive loop for building reflection zones.



(a) Retrieving data from a reflection zone map.

(b) Determining location of point relative to a reflection zone.

Figure B.6: Interfaces of the reflection zones during ray tracing.

Relevant Geometric Structures The technique can be used only with plates which can be single or connected.

Relevant rays for the reflection zones are listed in table B.2. Reflection zones can only be used with rays where the first and subsequent interactions are plate reflections.

Table B.2: Reflection Zone Rays.

Primary and Secondary Rays	**Tertiary Rays**
RP, RP-RP, RP-DE, RP-RC, RP-DC	RP-RP-RP

B.4.3 Reflection Zones in SuperNEC-UTD

In SuperNEC, the reflection optimisations consist of 2 parts.

1. Recursively determine the reflection zones and the components in the reflection zones.
2. Ray trace only into the reflection zones.

Recursively Determine the Reflection Zones A recursive algorithm is used to construct all orders of reflection zones (see figure B.5). The algorithm is passed a list of all the plates and a single segment. The image of the segment is calculated. Using the image all the reflection zones are constructed for the plates in the list. Constructing a reflection zone means, determining the planes which define the semi-infinite volume shown in figure B.4(b) and determining which components fall inside that volume.

For each plate that is visible in a reflection zone, the next higher order reflection zone can be built. To build a higher order reflection zone the visible plates in a reflection zone are fed back into the algorithm. For higher order zones the image of the image is determined. This procedure is repeated for all segments.

Ray-Tracing The 2 methods of accessing and using the reflection zones are illustrated in figure B.6. Reflection zones are stored in a map. The map is encapsulated in a class to provide an efficient interface. A map is an array where the index need not be numerical. Also the data type stored in a map can be defined by the user. In the reflection zones, the index into the map consists of 3 items: a segment, a plate and the order of reflection.

The first method of using a reflection zone is shown in figure B.6(a). Given a segment, a plate and the order of reflection, the map returns the reflection zone (a list of planes defining

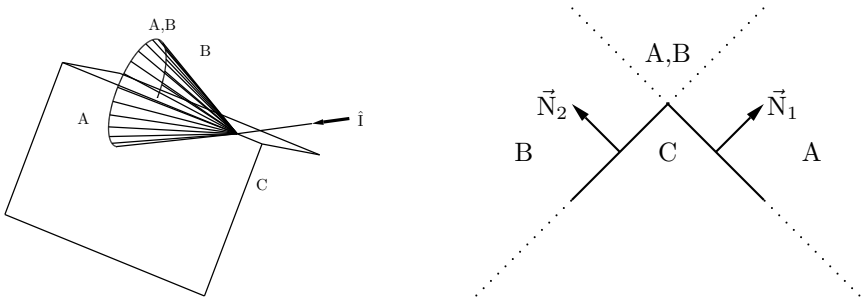
the semi-infinite volume) and the list of components illuminated by a reflection. The list of planes are used to determine if a point lies inside a reflection zone.

Reflection zones can be queried directly to determine if a point lies in the reflection zone. This is shown in figure B.6(b). Again a segment, a plate and the order of reflection are used to retrieve the reflection zone. A point is also passed to the zone to determine if the point is located within the reflection zone.

B.5 Diffraction Zones

In this section path finding optimisations for diffraction rays are explained. A diffraction zone is a spatial region defined at a wedge (an edge common to 2 plates). The spatial region contains all the plates and cylinders which are illuminated by a ray diffracting on the wedge. Using a diffraction zone geometrically invalid rays are quickly eliminated.

B.5.1 Principle of Operation



(a) Three dimensional diffraction zone. Adapted from [10].

(b) Edge-on view of a diffraction zone.

Figure B.7: Diffraction zones (3D and 2D).

Using the theory of diffraction as described by Keller [3], for each wedge the spatial region is determined which is illuminated by a ray diffracting on the edge. The spatial region is called a diffraction zone. Figures B.7(a) and B.7(b) show a 3 and a 2 dimensional illustration of a wedge. Figure B.7(b) shows the edge-on view of figure B.7(a).

In figure B.7(a), a ray is incident on an edge which results in infinitely many diffracted rays which form a cone around the wedge [10]. Depending on the location of the incident ray, the diffracted ray is limited to 2 spatial regions: the ray can either be above the wedge as shown in figure B.7(a) or below the wedge.

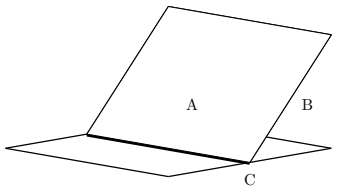
In the edge-on view (figure B.7(b)) the incident ray (not shown) is in the region pointed to by the normals of the plates ($\vec{N}_1$ and $\vec{N}_2$) [2]. The diffracted ray can only reach regions A, and B (ie. the union $A \cup B$).

If the incident ray is below the wedge in figure B.7(b), the diffracted ray can only reach region C. Diffraction zone C is defined by the *intersection* of the semi-infinite volumes defined by the 2 plates. In this case the semi-infinite volume is on the side *not* pointed to by the normals.

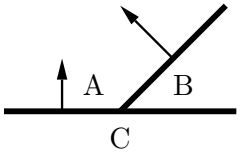
For a general wedge 2 diffraction zones need to be determined, ie. the union of regions A and B on the outside and region C on the inside of the wedge. If the incident ray is in region A or B the diffraction ray will also be in region A or B. Similar for region C. For polyhedrons only the diffraction zone outside of the polyhedron is determined.

Figure B.8 illustrates another possible wedge arrangement. Here a plate is joined to the surface of a second plate. Diffraction can only occur if the incident ray is located in either

[2]The normals of the plates must either both point into or outside the diffraction zone. In this case the normals point into the spatial region.



(a) 3 dimensional diffraction zone, for surface connected plates.



(b) Edge-on view of the 3 dimensional diffraction zone.

Figure B.8: Diffraction zones (3D and 2D) for surface connected plates.

region A or B. A ray in region C could only reflect off the plate. An incident ray in region A can only be diffracted to region A. Similar for region B.

B.5.2 Optimisation

Diffraction zones reduce the search space in 2 ways. Firstly to determine if 2 points can be coupled by an edge diffraction. Secondly a diffraction zone stores the components which are illuminated by a ray diffracting off an edge.

Edge diffracted rays can only reach points inside the diffraction zone. If an observation point lies outside the diffraction zone, then the ray does not need to be traced. If the source segment lies in region A in figure B.7(b) and the observer lies in region C, then no diffraction is possible. However if the source and observer lie in region A or B then the source can be coupled to the observer by a diffraction.

A diffracted ray in region C, can only diffract or reflect off components in region C (Similar for region $A \cup B$.). Thus components in $A \cup B$ are eliminated from path finding when the incident ray lies on region C. This reduces the search space for path-finding.

Relevant Geometric Structures Diffraction zones can only be used at an edge where either 2 plates are joined or where an edge is connected to the surface of another plate.

Relevant rays for the diffraction zones are listed in table B.3. Any ray which includes a plate diffraction can be optimised using the above technique.

Table B.3: Relevant rays for diffraction zones.

Primary and Secondary Rays		**Tertiary Rays**
DE-x	**x-DE**	
DE, DE-RP, DE-RC, DE-DC, DE-DE	RP-DE, RC-DE, DC-DE	DE-RP-RP, DE-RP-DE, DE-DE-RP, DE-DE-DE

B.5.3 Diffraction Zones in SuperNEC-UTD

The diffraction zones are implemented in 4 steps in SuperNEC.

1. Identify common edges.
2. Build the diffraction zone, ie. the spatial region.
3. Identify the components in the diffraction zone.
4. Trace diffraction rays only in the spatial region of the wedge from which the rays originate.

Identify common edges. All plates and their edges are examined to determine if they are connected to another edge or to the surface of another plate. This is explained in section B.3.3.

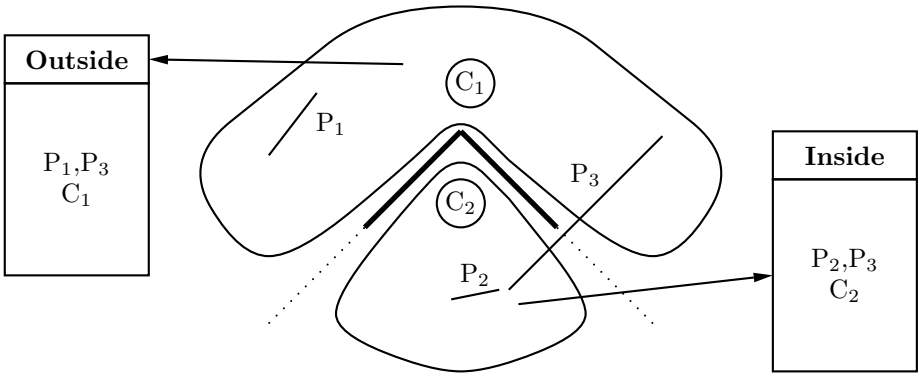


Figure B.9: Two diffraction zones illustrated for the 2D case.

Build the diffraction zone. For each plate of a wedge, the plane equation is determined. The equation consists of the vector normal to the plane and the distance of the plane from the origin. The 2 plane equations of a wedge define the diffraction zones.

Identify the components in the diffraction zone. Using the planes, the location of a component relative to the wedge is determined. It is possible to determine the location of a point relative to the diffraction zone, ie. either inside or outside the zone. Each component can lie either inside or outside a diffraction zone.

The outline of a component is used to determine the side of the wedge on which the component lies. For a plate the outline consists of the vertices of the plate. For the cylinder a bounding box is constructed. The bounding box consists of 8 vertices. To determine if a component lies on the inside of a diffraction zone, at least one vertex must lie in the inside. This is done using the dot product. For each diffraction zone a vector stores the visible components. If a component lies in both the inside and outside diffraction zone of a wedge then this component is listed in both vectors.

Figure B.9 shows a wedge and the components associated with the inside and outside diffraction zones. The blobs represent the semi-infinite spatial regions on either side of the wedge. The plates (shown by the bold lines) making up the wedge are used to define the spatial region. Components are categorised into either spatial region by determining on which side of the planes of the plates (shown by the dotted lines) they are located. The boxes labelled *inside* and *outside* in figure B.9 symbolise the vectors which store pointers to the components in each diffraction zone.

Ray tracing. During path finding both the source and the observer points of a diffraction ray must lie on the same side of a wedge. The source and observer can be the location of a segment, a far field point or a point on a component (that is a point of reflection or diffraction). A ray incident on an edge can only diffract to points or components that lie on the same side of the wedge as the incident ray.

The diffraction zone is used in the following ways

- Determine if a point is located in a diffraction zone.
- Given a point, return the components on that side of the wedge where the point is also located. In figure B.9 either the outside or the inside list of components is returned.
- Given a component eg. C_1 in figure B.9 return all the other components which are on the same side of the wedge as that component, ie. return the outside list in the example.

Errors

The causes of the errors are illustrated in figure B.10. When using the diffraction zones the diffracted rays can only be in the same spatial region as the incident ray. In the figure the

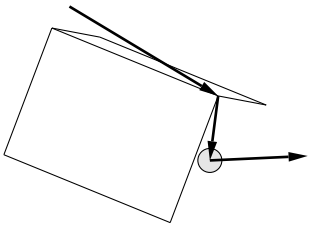


Figure B.10: Corner diffracted ray that propagates underneath the wedge.

Table B.4: Hardware and software used during the tests.

Hardware	
Processor	Intel(R) Pentium(R) 4 CPU [11]
Processor Frequency	3.80 GHz
Random Access Memory (RAM)	1.00 GB
Software	
Operating System (OS)	Microsoft Windows XP Professional
OS Version	Version 2002, Service Pack 2
SuperNEC Version	2.9
SuperNEC Compile Configuration	WIN32 Release
Compiler	Microsoft Visual C++ 6.0

incident ray strikes the corner of the wedge and propagates underneath the wedge. The way the diffraction zones are defined using the plane equations of the plates means that corner diffracted rays which propagate into the opposite region are eliminated.

As described earlier in section B.3.4 corner diffracted rays decay faster than edge diffracted rays [10]. The error measures and plots of the radiation plots show that corner rays can be neglected and that the resulting radiation patterns closely match the original radiation patterns.

B.6 Combined Performance

In the previous sections the optimisation techniques where presented. This section looks at the combined performance of the 3 optimisations. Two sets of tests examine the performance of the optimisations against the original software when the model size increases and when the number of segments increases. An overview of the tests is given in section B.6.1. Then the empirical results are presented and analysed.

B.6.1 Overview of the tests

The aims of the tests are to determine how the optimisation techniques perform collectively and how they perform when the geometry of the models is diverse. The models consist of polyhedrons, open structures (two or more connected plates), single plates and cylinders. Polyhedrons can be optimised with back face culling. Wedges found in open structures and polyhedrons are used with diffraction zones. Reflection zones work with all plates. In the next paragraphs specific information is given on the test models, radiation patterns and the relevant ray-types.

Hardware and software for the tests. All tests are run using the hardware and software described in table B.4. Table B.4 is for the most part self-explanatory. SuperNEC compile configuration in the 2nd last line of the table indicates that the compiler optimisations are used and debugging is disabled.

Test models. Details of the test models are given in table B.5. The 8 models used to examine the performance of the optimisations with increasing model size are listed in table B.5(a). Each model consists of a varying number of plates. The plates are arranged into polyhedrons and open structures. For polyhedrons and open structures the number in

Table B.5: Models used for combined path finding tests.

(a) Increasing model size.

	Models (increasing size)							
	1	2	3	4	5	6	7	8
Frequency	300 MHz							
Segments	7 segments							
Plates	7	14	21	28	35	42	49	56
Polyhedrons	1 (6)	1 (6)	1 (20)	1 (6)	2 (12)	2 (32)	3 (38)	3 (32)
Open structures	0	2 (6)	0	1 (19)	1 (19)	2 (6)	2 (6)	4 (20)
Single plates	1	2	1	3	4	4	5	4
Cylinders	1	1	2	2	3	3	4	4

(b) Increasing number of segments.

	Models (increasing number of segments)							
	1	2	3	4	5	6	7	8
Frequency	300 MHz							
Segments	7	12	17	22	27	32	37	42
Plates	34							
Polyhedrons	1 (20)							
Open structures	2 (10)							
Single plates	4							
Cylinders	3							

Table B.6: Radiation patterns for the test models. The column *degrees* gives the number of sample points taken in the azimuth or in the elevation planes.

	θ			ϕ		
Plane	Start	End	Degrees	Start	End	Degrees
XY Plane	90.0°	90.0°	1	0.0°	360.0°	181
XZ Plane	0.0°	360.0°	181	0.0°	0.0°	1
YZ Plane	0.0°	360.0°	181	90.0°	90.0°	1

brackets is the number of plates used for those structures. The number of cylinders is also specified.

The 8 models used to investigate the optimisation techniques with increasing number of segments are given in table B.5(b). The number of components in the models remain constant but the number of segments varies from 7 to 42.

The radiation patterns in table B.6 are calculated using the original and optimised programs. During the execution of the programs the path finding times and the number of rays are recorded.

Ray Types The tests in this section only use the rays listed in table B.7. Each optimisation technique can only be used with particular geometric structures (polyhedrons, wedges or plates). This means that only certain ray types can be optimised.

In this first part of table B.7 all the ray-types for the tests are listed. The next three sections list the ray-types that can only be optimised with back face culling, reflection zones and diffraction zones. Some rays can be optimised using more than one technique.

Speed-Up The speed-up is given by the ratio of the original run time over the run time of the optimised software [12].

$$\text{Speed up} \;=\; \frac{\text{original time}}{\text{optimised time}} \tag{B.1}$$

Error Measure The absolute error of the total power radiated is calculated using [13]

$$\text{ae} \;=\; |\text{x} - \text{y}| \tag{B.2}$$

where x is the total power in the far field obtained using the optimisation techniques and y is the value obtained using the original software. x and y and the absolute error are given in dB.

Table B.7: Relevant ray-types for combined tests.

All Ray-Types		
Component	**First Order**	**Second Order**
Plates	RP,DE	RP-RP, RP-DE, DE-RP, DE-DE
Combined		RP-RC, RP-DC, DE-RC, DE-DC, RC-DE, DC-DE
Ray-Types for Back Face Culling		
Component	**First Order**	**Second Order**
Plates	RP,DE	RP-RP, RP-DE, DE-RP, DE-DE
Combined		RP-RC, RP-DC, DE-RC, DE-DC
Ray-Types for Reflection Zones		
Component	**First Order**	**Second Order**
Plates	RP	RP-RP, RP-DE
Combined		RP-RC, RP-DC
Ray-Types for Diffraction Zones		
Component	**First Order**	**Second Order**
Plates	DE	RP-DE, DE-RP, DE-DE
Combined		DE-RC, DE-DC, RC-DE, DC-DE

The mean absolute error is given by

$$m_{ae} = \frac{1}{N}\sum^{N} ae_i \tag{B.3}$$

where ae_i is the absolute error in the radiated power at a single far field point, N is the number of far field locations and m_{ae} is the mean absolute error.

B.6.2 Empirical Results for Increasing Model Size

The results for the 8 models in table B.5(a) which increase in size are presented here.

Table B.8: Path finding speed-up factors for the individual optimisation techniques and combined optimisation techniques.

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**	**Model 6**	**Model 7**	**Model 8**
BFC	1.27	1.13	1.61	1.06	1.06	1.29	1.28	1.20
RZ	1.04	1.05	1.05	1.06	1.04	1.05	1.05	1.04
DZ	1.11	1.22	1.31	1.36	1.35	1.53	1.45	1.48
All	1.53	1.47	2.76	1.59	1.54	2.27	2.11	1.99

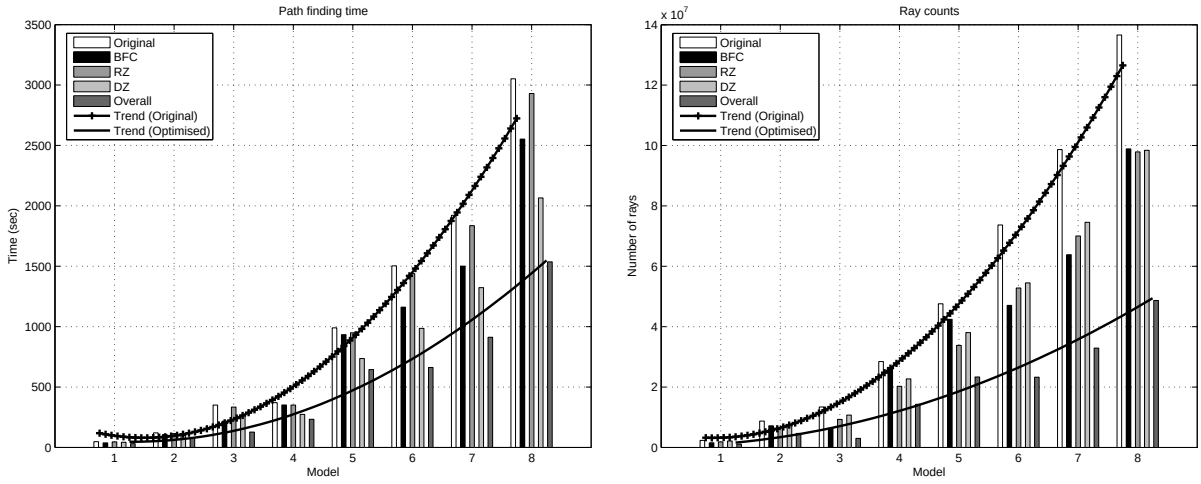
The speed-up factors for each individual optimisation and for the combined optimisations are given in table B.8. Overall the path finding time is reduced up a factor between 1.53 and 2.27.

Table B.9: Search space reduction for the individual optimisation techniques and combined optimisation techniques.

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**	**Model 6**	**Model 7**	**Model 8**
BFC	1.52	1.22	2.10	1.07	1.12	1.57	1.55	1.38
RZ	1.27	1.33	1.44	1.40	1.41	1.40	1.41	1.40
DZ	1.08	1.17	1.25	1.25	1.25	1.35	1.32	1.39
All	2.13	1.88	4.49	1.99	2.04	3.17	3.00	2.81

In table B.9 the reduction in the number of rays shot during the experiments due to each optimisation relative to the original program is given. Using the combined optimisation techniques the search space is reduced by between 1.88 and 3.17.

The run times are plotted in figure B.11(a) and the number of rays in figure B.11(b). A group of 5 bars represents the results for one model. The results for the original program, the 3 individual techniques and the combination are shown. Both figures indicate that the combined optimisation techniques halve the path finding time and search space. However the trend lines indicate that the run time and search space continue to increase quadratically when using the optimisations. On average the optimised program is 1.90 times faster than the original.



(a) Run times.

(b) Search space.

Figure B.11: Run times and search space for the 8 models with increasing model size.

In general DZ have the largest speed-up. As the model size increases there are more wedges which are used to build DZ. In the original program the increasing number of wedges causes an exponential growth in the number of diffraction rays. DZ limit the number of diffracted rays at each wedge thus limiting the exponential ray growth. This is shown by the reduced run times for DZ in figure B.11 and by the speed-ups in table B.8. On average the DZ speed-up is about 1.35 which is a 26 % reduction in the run time.

BFC is less effective than DZ because the technique only identifies plates of polyhedrons visible to segments. The speed-up results vary with the number of polyhedrons (listed in table B.5(a)). Models with a large number of polyhedrons have a high BFC speed-up. The average speed-up using BFC is 1.24 which is a 19 % reduction in the run time.

RZ are the least effective in reducing the run time. The technique can only be used with reflections that occur consecutively and that start from a segment. RZ are less dependent on the geometric structure of a model because the technique works with all plates. The RZ speed-up is therefore fairly constant across all models. The average speed-up is 1.05 which is a 5 % reduction in the run time.

Errors for Increasing Model Size

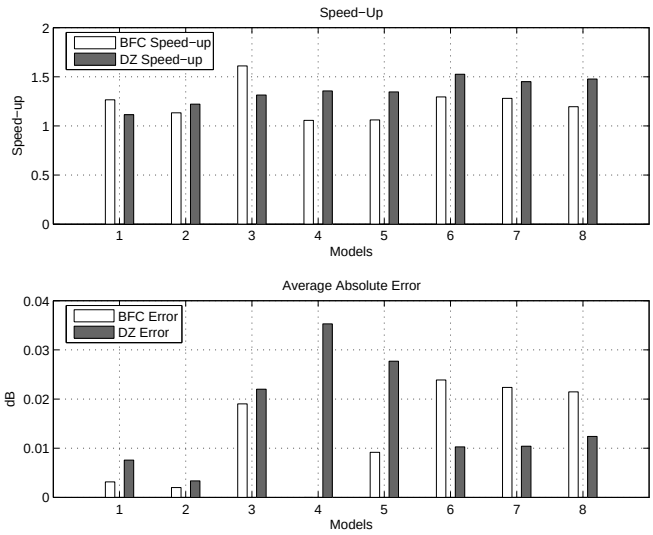
Errors in the radiation patterns due to back face culling and diffraction zones are discussed in this section.

Table B.10: Errors when using back face culling (BFC), diffraction zones (DZ) and all optimisations. m_{ae} (dB) is the mean absolute error.

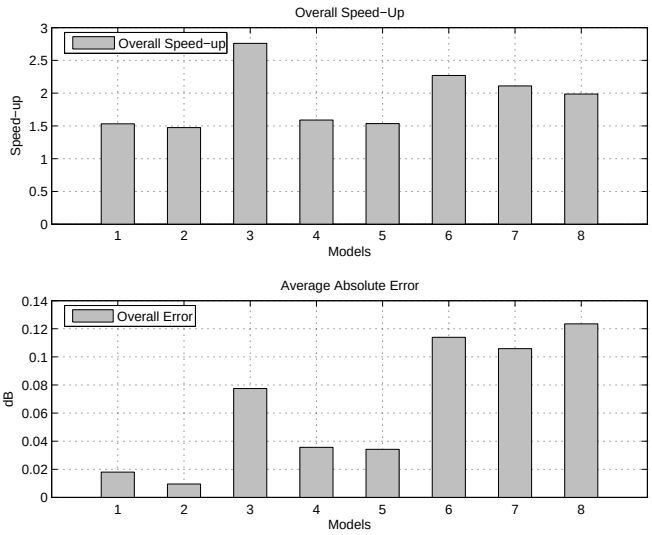
	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**	**Model 6**	**Model 7**	**Model 8**
BFC								
m_{ae}	3.15×10^{-3}	1.99×10^{-3}	1.90×10^{-2}	0	9.15×10^{-3}	2.39×10^{-2}	2.24×10^{-2}	2.15×10^{-2}
DZ								
m_{ae}	7.59×10^{-3}	3.35×10^{-3}	2.20×10^{-2}	3.53×10^{-2}	2.77×10^{-2}	1.03×10^{-2}	1.04×10^{-2}	1.24×10^{-2}
All								
m_{ae}	1.80×10^{-2}	9.54×10^{-3}	7.75×10^{-2}	3.57×10^{-2}	3.42×10^{-2}	1.14×10^{-1}	1.06×10^{-1}	1.23×10^{-1}

Table B.10 lists the errors for the 8 test models when only BFC, when only DZ and when all optimisations are used. For each optimisation the mean absolute error (m_{ae}) is given. The values in the table are the overall errors in all 3 radiation patterns for each of the 8 models.

The mean absolute errors for back face culling and the diffraction zones are shown in the bottom graph in figure B.12(a). The top graph in figure B.12(a) shows the speed-ups for back face culling and the diffraction zones. The 2 bar graphs show no correlation between the error and the speed-up. Figure B.12(b) shows the speed-up (top graph) and the mean



(a) Speed-up and errors for BFC and DZ.



(b) Overall speed-up and overall errors.

Figure B.12: Speed-up and errors for test models (increasing model size).

error (bottom graph) when all optimisations are used. The relationship between the speed-up and the error is difficult to determine. Figure B.12(b) shows that when the speed-up is high the error tends also to be high. The error is larger for models with more components. In larger models there will be more geometrical structures which cause the errors.

To give a feel for the errors in table B.10 and figure B.12, the XZ-elevation plane ($\theta = -180°..180°, \phi = 0°$) radiation pattern for model 8 is plotted in figure B.13. The figure shows the radiation pattern using all 3 optimisation techniques. In the figure the radiation pattern from the original program is also shown. The errors are very slight and the graphs almost overlap. The errors are between $\theta = -30°$ and $\theta = 30°$.

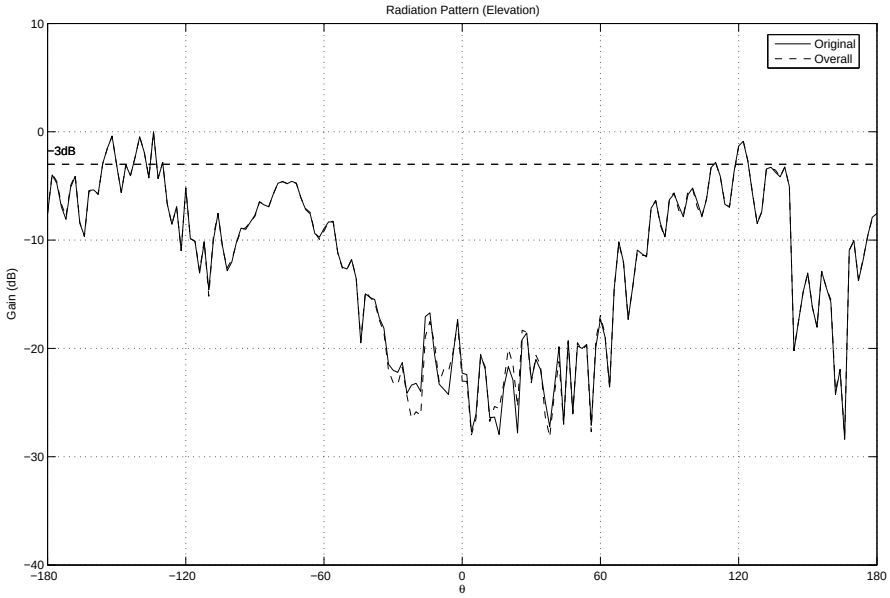


Figure B.13: Radiation pattern ($\theta = -180°..180°, \phi = 0°$) for model 8 using all optimisations. m_{ae}=2.28×10^{-1} dB

B.6.3 Empirical Results for Increasing Number of Segments

The results for the 8 models in table B.5(b) with varying number of segments are presented.

Table B.11: Path finding speed-up factors for the individual optimisation techniques and combined optimisation techniques.

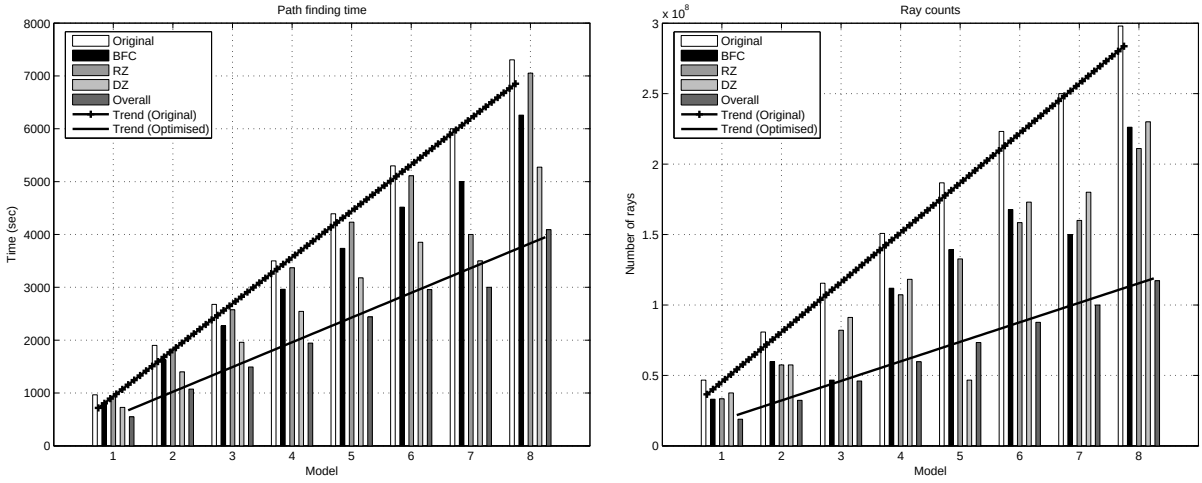
	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**	**Model 6**	**Model 7**	**Model 8**
BFC	1.19	1.17	1.18	1.18	1.18	1.17	1.17	1.17
RZ	1.04	1.04	1.04	1.04	1.04	1.04	1.04	1.04
DZ	1.33	1.36	1.37	1.38	1.38	1.38	1.39	1.39
All	1.76	1.77	1.79	1.80	1.80	1.79	1.79	1.79

The path finding speed-up factors are given in table B.11 for each optimisation technique and when using all techniques. The reduction in the number of rays using the optimisations

Table B.12: Search space reduction for the individual optimisation techniques and combined optimisation techniques.

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**	**Model 6**	**Model 7**	**Model 8**
BFC	1.41	1.35	2.47	1.35	1.34	1.33	1.67	1.32
RZ	1.40	1.41	1.41	1.41	1.41	1.41	1.56	1.41
DZ	1.24	1.41	1.27	1.28	4.00	1.29	1.39	1.30
All	2.47	2.50	2.51	2.52	2.55	2.55	2.50	2.54

is given in table B.12.



(a) Run times.

(b) Search space.

Figure B.14: Run times and search space for the 8 models with increasing number of segments.

The run times are shown graphically in figure B.14(a) and the number of rays are plotted in figure B.14(b). The trend lines indicate that path finding time and the number of rays increase linearly with the number of segments. Using the optimisations the program is on average 1.79 faster than the original. This is a 44 % reduction in the run-time.

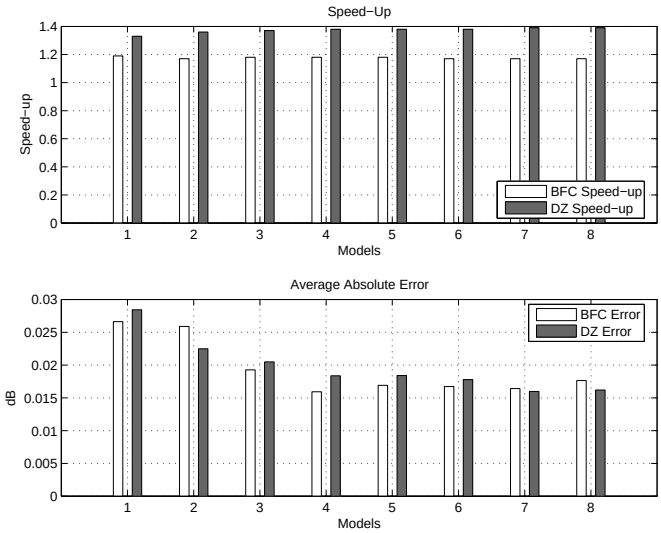
As explained in the previous section, DZ are the most useful in reducing the run-time and search space. The average speed-up factor for DZ is 1.37 which is a 27 % reduction in the run-time. BFC has the next highest speed-up (the average is 1.18 which is a 15 % reduction in the run-time). RZ reduce the run time by on average 1.04 which is approximately a 4 % reduction in the run-time.

Errors for Increasing Number of Segments

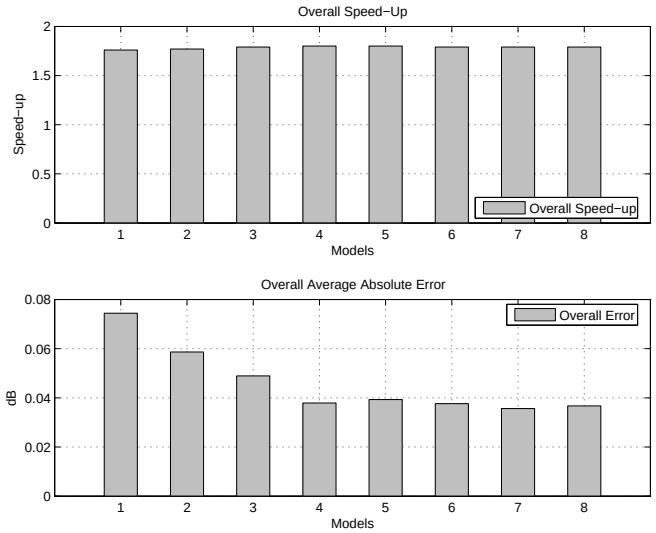
Errors in the radiation patterns due to back face culling and diffraction zones are discussed in this section. Table B.13 gives the mean absolute error (m_{ae}) for the 8 models using

Table B.13: Errors when using back face culling (BFC), diffraction zones (DZ) and all optimisations. m_{ae} (dB) is the mean absolute error.

	Model 1	**Model 2**	**Model 3**	**Model 4**	**Model 5**	**Model 6**	**Model 7**	**Model 8**
BFC								
m_{ae}	2.66×10^{-2}	2.59×10^{-2}	1.93×10^{-2}	1.59×10^{-2}	1.69×10^{-2}	1.67×10^{-2}	1.64×10^{-2}	1.76×10^{-2}
DZ								
m_{ae}	2.84×10^{-2}	2.25×10^{-2}	2.05×10^{-2}	1.84×10^{-2}	1.84×10^{-2}	1.78×10^{-2}	1.60×10^{-2}	1.62×10^{-2}
All								
m_{ae}	7.44×10^{-2}	5.86×10^{-2}	4.89×10^{-2}	3.79×10^{-2}	3.93×10^{-2}	3.76×10^{-2}	3.57×10^{-2}	3.67×10^{-2}



(a) Speed-up and errors for BFC and DZ.



(b) Overall speed-up and overall errors.

Figure B.15: Speed-up and errors for test models (increasing number of segments).

back face culling, diffraction zones and the combination of the 3 optimisations. The mean absolute errors for back face culling and diffraction zones are plotted in the bottom graph of figure B.15(a). The speed-up for the 2 techniques is shown in the top graph. On the right side in figure B.15(b) the speed-up using all 3 techniques is shown (top graph). Below that the overall mean absolute errors are shown.

The errors in figure B.15 decrease with increasing number of segments while the speed-up remains fairly constant. As the number of segments increases the greater number of rays traced compensates for the rays that are erroneously eliminated by back face culling and the diffraction zones. For the 8 models the error is not correlated to the speed-up. The error is largely a function of the geometry of the models.

The 2D radiation pattern ($\theta = -180°..180°, \phi = 90°$) in figure B.16 (on page B.50) shows the difference between the original program and the optimised program. The radiation pattern is for model 1 which has the largest error out the eight models. The errors are very slight and the original and optimised graphs almost overlap. In the figure the errors are around $\theta = -130°$ and $\theta = 130°$.

B.6.4 Discussion of Empirical Results

Path finding optimisations work equally well with increasing model size and increasing number of segments. Increasing model sizes lead to exponentially increasing run times. When increasing the model size and using the optimisations, the increase in path finding time remains quadratic although the gradient is smaller and the run time is reduced. A change in the number of segments causes a proportional change in the run time.

Although the optimisations introduce errors, the results are acceptable given the speed-ups. The radiation patterns indicate that the optimised outputs closely follow the original. Error and speed-up are not correlated and often the errors are localised to a small range of ϕ or

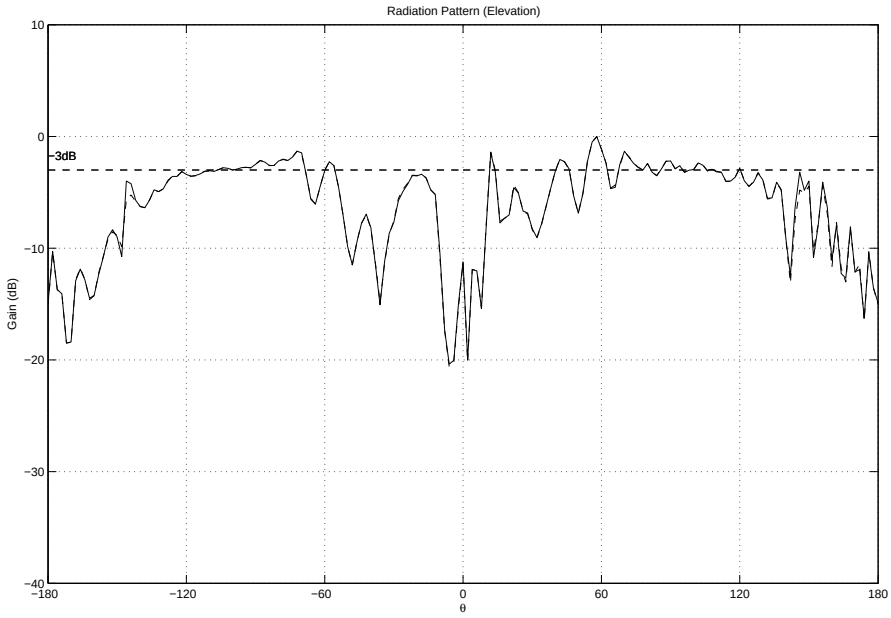


Figure B.16: Radiation pattern ($\theta = -180°..180°, \phi = 90°$) for model 1 using all optimisations. m_{ae}=9.04×10^{-2} dB

θ values. Errors are caused by the geometric structure of the models which leads to the localisation of the errors.

B.7 Tabular Overview of the Optimisation Techniques

In this section a comparative summary is given of the 3 optimisation techniques. Table B.14 is divided into 4 sections: overhead, geometric structures, relevant ray-types and performance.

B.7.1 Overhead

The overhead indicates if the time to build and the memory used to store the data structures for the optimisation techniques is dependent on the frequency or the model size.

Frequency dependent Indicates if the time and memory overhead increases with the simulation frequency. As frequency increases the number of segments tends to increase. Both BFC and RZ use the location of each segment to determine the visible components. Therefore the number of data structures used in BFC and the number of RZ are a function of the number of segments. DZ are only built for each wedge and therefore the number of DZ is not dependent on the frequency.

Model size dependent Indicates if the time and memory overhead increases with the model size. Model size refers to the number of plates and cylinders in the model. As the number of components increases, all optimisation techniques will store more components in their data structures. In general more reflection and diffraction zones are created as the number of plates increases. For BFC, as more components are added, the number of components *seen* by a segment increases which increases the memory used.

B.7.2 Geometric Structures

Geometric structures refers to the type of geometry which is required by the optimisations.

Table B.14: Overview of the path finding optimisations.

	Back face culling	**Reflection Zones**	**Diffraction Zones**
		Overhead	
Frequency Dependent	Yes	Yes	No
Model Size Dependent	Yes	Yes	Yes
		Geometric Structures	
		Required	
Cylinders	No	No	No
Plates	Yes	Yes	Yes
		Illuminated	
Cylinders	No	Yes	Yes
Plates	Yes	Yes	Yes
		Plate structures	
Polyhedrons	Only	Yes	Yes
Open Structures	No	Yes	Wedges
Wedges	No	Yes	Yes
Single Plates	No	Yes	No
		Relevant Ray-types (plates)	
Direct	No	No	No
First reflections	Yes	Yes	No
First diffractions	Yes	No	Yes
Consecutive second and higher reflections	No	Yes	No
Arbitrary second and higher reflections	No	No	No
Second and higher diffractions	No	No	Yes
		Performance	
Speed-up	medium	low	high
	15 %	5 %	30 %
Absolute Error	Yes	No	Yes
	Small	–	Small
	1.60×10^{-2} dB	–	1.79×10^{-2} dB

Required These components must be present for the optimisations to work. All optimisations work with plates. No cylinders are required.

Illuminated Indicates which techniques can determine whether a cylinder or a plate is illuminated. Both RZ and DZ can determine if a cylinder is illuminated. For example RZ can be used to determine which cylinders are seen by an RP-RC ray (ie. a plate reflection followed by a cylinder reflection). DZ can determine which cylinders are illuminated by, eg. a DE-RC ray. All techniques can determine if a plate is illuminated.

Plate structures Each optimisation works with certain geometric structures. Reflection zones work with any structure consisting of one or more plates. BFC works only with polyhedrons. DZ also work with polyhedrons and with open structures where 2 plates join to form a wedge. The edges of an open structure which are not connected cannot be used with DZ.

B.7.3 Relevant Ray Types

Each optimisation can only be used with certain plate ray types.

Direct No path finding technique optimises line of sight (direct) rays.

First reflection Indicates which technique optimises rays which start with a plate reflection.

First diffraction Indicates which technique optimises rays which start with an edge diffraction.

Consecutive second and higher reflections Indicates if successive plate reflections after the first reflection are optimised. For example, double reflection (RP-RP) and triple reflection (RP-RP-RP) are optimised using RZ but not with BFZ or DZ.

Arbitrary second and higher reflections Indicates if an arbitrary order reflection can be optimised, eg. the reflection in a DE-RP or RC-RP rays, cannot be optimised by any technique.

Second and higher diffractions Indicates if an arbitrary order edge diffraction can be optimised, eg. the diffraction in RP-DE, DC-DE or DE-RC rays can be optimised using DZ.

B.7.4 Performance

Performance refers to the speed-up and error due to the techniques.

Speed-up The first row compares the speed-up of the 3 techniques. DZ in general have the highest speed-up factors, followed by BFC and RZ have the smallest speed-up. The second row provides an quantitative indication of the reduction in the original run time achieved by the techniques, eg. using DZ a 30 % reduction in the original run time can be expected.

Error The first row indicates which technique introduces errors. The next 2 rows provide a qualitative and a quantitative indication of the mean absolute error. The values give an indication of the magnitude of error that can be expected.

B.8 Conclusion

Three optimisation techniques for path finding are presented: back face culling, reflection zones and diffraction zones.

Collectively the optimisations reduce the run time by about half. The techniques work well with plates and models which consists mostly of plates. The techniques can in some situations determine which cylinders are illuminated.

References

[1] SuperNEC. Poynting Antennas and Electromagnetics. Last accessed 15 June 2006. [Online]. Available: http://www.supernec.com

[2] A. Fourie and D. Nitch, "SuperNEC: Antenna and indoor propagation simulation program," *IEEE Antennas and Propagation Magazine*, vol. 42, no. 3, June 2000.

[3] J. Keller, "Geometrical theory of diffraction," *Journal of the Optical Society of America*, vol. 52, no. 2, pp. 116–130, 1962.

[4] R. G. Kouyoumjian and P. H. Pathak, "A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface," *Proceedings of the IEEE*, vol. 62, no. 1, Nov. 1974.

[5] A. Glassner, "Space subdivision for fast ray tracing," *IEEE Computer Graphics and Applications*, vol. 4, no. 10, pp. 15–22, Oct. 1984.

[6] A. S. Glassner, Ed., *An Introduction to Ray Tracing.* Academic Press, 1989.

[7] R. Hartleb, "Optimised ray tracing for the SuperNEC implementation of the uniform theory of diffraction," MSc(Eng) Dissertation, University of the Witwatersrand, Johannesburg, 2006, Appendix A: Analysis of the run time of SuperNEC-UTD: a MoM electromagnetic simulation package.

[8] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice.* Addison-Wesley, 1990.

[9] M. Kimpe, H. Leib, O. Maquelin, and T. H. Szymanski, "Fast computational techniques for indoor radio channel estimation," *Computing in Engineering and Science*, vol. 1, no. 1, pp. 31–41, Jan–Feb 1999.

[10] D. A. McNamara, C. W. I. Pistorius, and J. A. G. Malherbe, *Introduction to the Uniform Theory of Diffraction.* Artech House, 1990.

[11] Intel pentium 4 processor. Intel Corporation. Last accessed 3 April 2006. [Online]. Available: http://www.intel.com/products/processor/pentium4/

[12] R. E. Bryant and D. R. O'Hallaron, *Computer systems: a programmer's perspective.* Upper Saddle River, NJ: Prentice Hall, 2003.

[13] S. C. Chapra and R. P. Canale, *Numerical methods for engineers.* McGraw-Hill, 2006.

Appendix C

Shadow Test Optimisations for SuperNEC-UTD

Abstract

The results of the SuperNEC-UTD shadow test optimisations using an octree are presented. An octree reduces the number of intersection tests for a ray to only those components which are in the vicinity of the ray. The theory of octrees and the software implementation are briefly discussed. One set of tests determine how effective the octree is in reducing the run time of SuperNEC-UTD. Another set of tests determine how well the octree shadow tests scale with model size and with the number of segments. During the tests radiation patterns are determined using the UTD method and the time and memory overhead, the run times and the number of intersections tests during the original and optimised programs are recorded. The shadow test optimisation using an octree is effective only for simple rays. Shadow time for higher order rays is reduced using the octree optimisation but the path finding time outweighs the shadow test time. For simple rays the overall speed-up factor is about 1.70 which is about a 41 % reduction in the run time. The shadow tests themselves for simple rays are about 8 times faster using the octree. This means the shadow time is reduced by about 88 %. For higher order rays the octree does not reduce the total run time (speed-up approximately unity) but the shadow test time is reduced by a factor of 5 on average. The octree shadow tests scale with model size and with the number of segments.

C.1 Introduction

SuperNEC-UTD uses the concept of rays to determine the propagation of high frequency electromagnetic waves from a radiating source (eg. an antenna) in the presence of a model (eg. aeroplanes, boats etc.) constructed with plates and cylinders (called components). The path travelled by a ray from the source to an observation point in the far field can include multiple reflections and diffractions off plates and cylinders.

Once a ray path has been determined from a source to an observer, shadow tests determine if the ray's path is intersected by another component. At present a brute force method is used in SuperNEC-UTD which checks the ray against every component in the scene to determine if the ray is shadowed. At best only one intersection test is done, if the ray intersects the first component. At worst, for a model consisting of n-components, n-intersection tests are done. The brute force shadow test method does not scale with model size. A detailed discussion of the problems with the original shadow algorithm are given in [1]

This report discusses shadow optimisations based on octrees. Similar to the way a grid on a map is used to locate objects, an octree is a data structure which sorts the information in a 3D volume, so that information at a particular point can be retrieved quickly. The octree improves the scalability of the software. Using an octree, the components which lie close to or in the path of a ray are determined quickly, and the number of intersection tests reduced to only those components.

The report is structured as follows. Section C.2 describes the principle behind the octree, how the octree is constructed, how rays are traced through the octree and how the intersection tests are optimised. The octree implementation used in SuperNEC-UTD is briefly discussed in section C.2.3. Then in section C.3 the tests that are run using the octree are presented. Following that, the empirical results of the tests are presented and discussed. These results include the overhead using the octree, the results for simple rays, the results for second order rays and the results for tertiary rays. The scalability of the octree shadow tests is examined in section C.4.

C.2 Introduction to Octrees

In this section the octree data structure and the implementation used in the shadow tests in SuperNEC-UTD are presented. For a detailed discussion on octrees and related data structures the reader is referred to [2, 3, 4, 5, 6, 7, 8, 9].

An octree is a geometrical data structure which is used to store and access information about a spatial region. Similar to the 2D grid used to quickly locate a geographical features on a map, an octree is an indexed 3D grid of a volume. Time to access information in the octree is independent of the size of the spatial region and of the amount of information stored. In SuperNEC-UTD the spatial region is the 3D volume and the information is the model made up of plates and cylinders. The octree is a time space tradeoff which means that detailed information is stored about the components in a model and this information is used to decrease the time required to run the shadow tests.

C.2.1 Constructing the Octree

The volume that is to be stored in the octree, is successively subdivided (see figure C.1). Initially the volume is divided into 8 sub-volumes called voxels. Depending on the density of components in a voxel, the voxel is recursively subdivided. The division stops when each voxel contains only a maximum number of components or when the maximum depth of the octree has been reached.

The 2D equivalent of an octree (called a quadtree [2]) is shown in figure C.2. Initially the 2D region is undivided (figure C.2(a)). Next, the region is divided into 4 sub-regions (figure C.2(b)). An octree is divided into 8 voxels. Empty regions or regions containing too few components are not divided any further. In figure C.2(c), the top left region is not subdivided. All the other regions contain too many components and are recursively

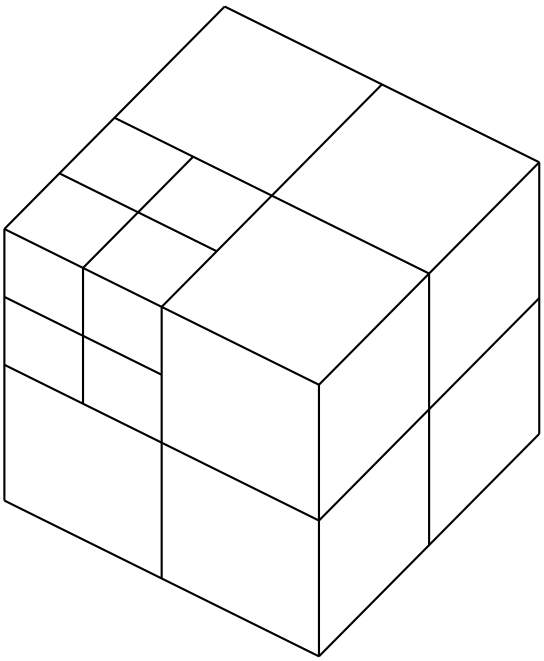
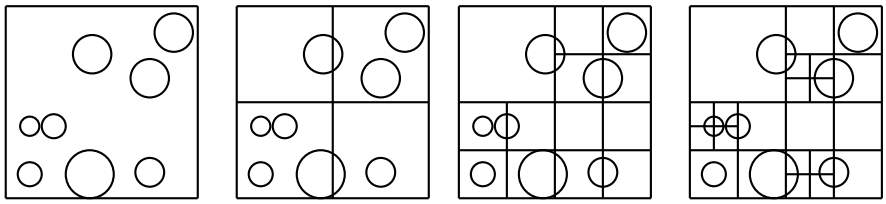


Figure C.1: Three dimensional view of the spatial subdivision used in an octree. Adapted from [2].



(a) Undivided. (b) Depth 1 (c) Depth 2 (d) Depth 3

Figure C.2: Building a quadtree. Adapted from [4]

divided. This process is continued until all regions contain only a maximum allowed number of components (figure C.2(d)). In the example, subdivision stops when a region contains at most one component.

C.2.2 Ray Tracing using Octrees

This section discusses how a ray is used to retrieve from the octree only those components which are in the vicinity of the ray. The input to the octree is a ray representing the path of a high frequency electromagnetic wave. The output is a list of components in the vicinity of the ray.

The k-d tree traversal algorithm described by Subramanian [10] is used to trace a ray through the octree. The algorithm determines if the ray intersects the volume defined by the octree. If the ray is outside of the octree volume, the ray is not shadowed and the test ends. Otherwise, a test is done to determine which of the 8 voxels the ray intersects. The intersected voxels are quickly determined because the octree is aligned parallel to the coordinate system and a fast intersection method [11] is used. If the intersected voxel is not subdivided, the list of components contained in that voxel is returned and the intersection tests are run. For subdivided voxels, the ray is again recursively split over the child voxels. The procedure is repeated for all voxels the ray intersects, until the ray intersects a component or until all voxels have been explored.

Figure C.3 shows ray tracing in the 2D equivalent of an octree. In figure C.3(a), a ray intersects the region. The ray is first clipped to the region which contains the model (fig-

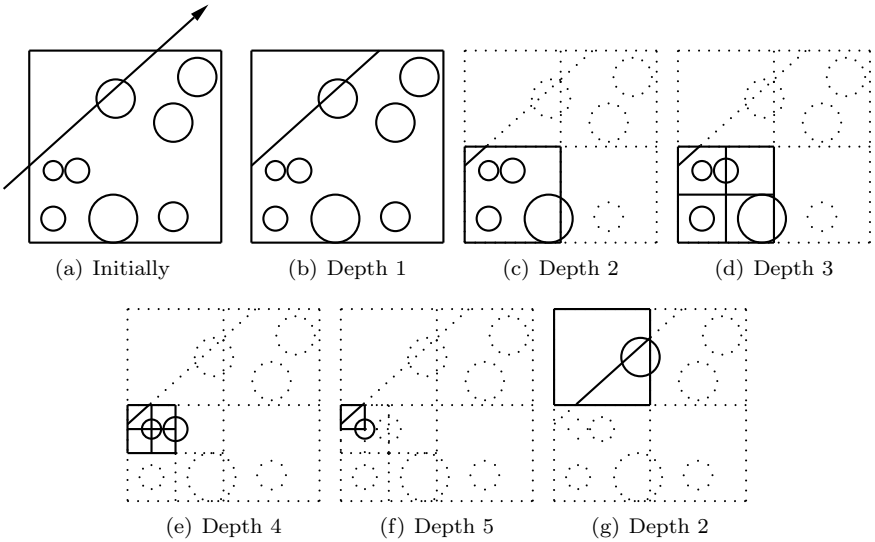


Figure C.3: Ray tracing in a quadtree.

ure C.3(b)). The ray intersects the lower left region of the quadtree. In figures C.3(c) to C.3(f), the quadtree is recursively traversed until the a region is found which is not subdivided. Intersection tests are run for the components in this region. If no intersections are found, the algorithm ascends and then descends the quadtree, to the next undivided region. In figure C.3 the quadtree is ascended to the second level in figure C.3(c), before the algorithm descends to the top left region shown in figure C.3(g). An intersection occurs and the algorithm ends.

Scalability using the Octree

The shadow test algorithm which uses an octree scales well with model size. The octree returns a list of components which lie close to the ray. This list contains only a fraction of the total number of components. Adding more components to the model will increase the size of the octree. A larger model will be subdivided using more voxels. Shadow tests time is independent of model size. In section C.4 empirical results show how the octree shadow tests scale with model size.

C.2.3 Octree Class in SuperNEC

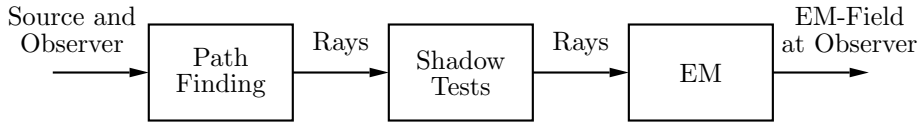
In figure C.4, block diagrams of the original and optimised programs are shown. The original software (figure C.4(a)) consists of 3 parts: path finding, shadow tests and electromagnetic (EM) calculations. The input are the source and observer points. The input to the shadow tests are all the geometrically valid rays from path finding. The shadow tests output all the non-blocked rays which are used to calculate the EM fields at the observer.

In figure C.4(b) the optimised software is shown. In the preprocessing stage the octree is constructed. For each ray passed from path finding, only the components which lie in the vicinity of a ray are retrieved and passed to the shadow tests.

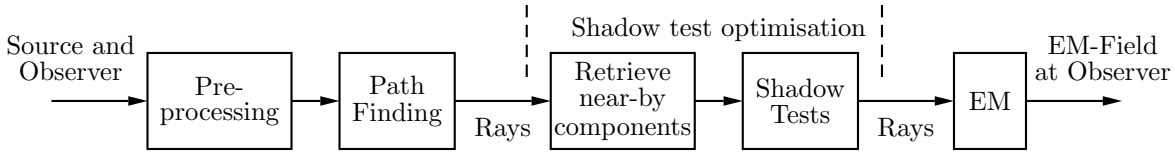
The implementation has 2 important interfaces.

1. A single function call to construct the octree.
2. A single function call to determine if a ray is shadowed or not.

When constructing the octree, the list of plates and cylinders making up the model is passed to the function. The recursive method described in section C.2.1 is used to build the octree. The user can set the maximum depth of the tree and the ideal number of components in a single voxel. A voxel is recursively subdivided if the voxel is not empty and the depth



(a) Block diagram of the original software.



(b) Block diagram of the optimised software.

Figure C.4: Block diagrams of SuperNEC-UTD.

Table C.1: Hardware and software used during the tests.

Hardware	
Processor	Intel(R) Pentium(R) 4 CPU [12]
Processor Frequency	3.80 GHz
Random Access Memory (RAM)	1.00 GB
Software	
Operating System (OS)	Microsoft Windows XP Professional
OS Version	Version 2002, Service Pack 2
SuperNEC Version	2.9
SuperNEC Compile Configuration	WIN32 Release
Compiler	Microsoft Visual C++ 6.0

of the tree has not reached the maximum depth. The ideal number of components is the maximum number of components that should be in a voxel. Division of a voxel stops as soon as the number of components in a voxel equals the ideal number of components. This automatically limits the depth of the octree. Consider the corner of a cube. At the corner 3 cube faces meet. If the ideal number is set to 3 and a voxel contains the corner of the cube, subdivision will immediately stop. The software will not attempt to divide the corner if the voxel contains 3 or fewer components.

A ray is passed to an instance of the octree, to retrieve the list of components that are within the voxels traversed by the ray. The traversal method is described in section C.2.2. Voxels are visited along the direction of the ray. As soon as a voxel is intersected, and it is not subdivided, the list of components in that voxel is retrieved and the intersection tests are done. The octree returns a list of components which is much smaller than the total number of components. The ray is compared against each component in the reduced list until an intersection is found or until all components have been tested. If no intersection is found the next voxel is visited. As soon as an intersection is found or when all voxels have been visited, the algorithm ends.

C.3 Testing the Octree in SuperNEC

In this section the results of tests using the octree in SuperNEC-UTD are presented. The aim of the tests is 1) to determine the overhead (in terms of memory and time) required to build the octree and 2) to provide a measure of the performance increase relative to the other parts of SuperNEC-UTD.

In the next section, an overview of the experiments is given. This is followed by section C.3.2 which presents and discusses the overhead results for all models. Sections C.3.3 to C.3.5 discuss the results of the shadow optimisations.

C.3.1 Overview of the Tests

Hardware and software for the tests. The octree tests are run using the hardware and software described in table C.1. The table is for the most part self-explanatory. Su-

Table C.2: Ray-types for the tests.

Simple Ray Tests			
Component	**Primary**	**Secondary**	**Tertiary**
Line of sight	direct ray		
Plate	RP, DE	–	–
Cylinder	RC, DC	–	
End cap	RL, DL	–	

Combined Ray Tests			
Component	**Primary**	**Secondary**	**Tertiary**
Line of sight	direct ray		
Plate	RP, DE	RP-RP, RP-DE, RP-RC, RP-DC, DE-DE, DE-RP, DE-RC, DE-DC	–
Cylinder	RC, DC	DC-DC	
End cap	RL, DL	DL-DL	

Tertiary Ray Tests			
Component	**Primary**	**Secondary**	**Tertiary**
Plate	–	–	RP-RP-RP, DE-DE-DE, DE-DE-RP, DE-RP-DE, DE-RP-RP

Table C.3: Test models.

Model Set 1 (increasing number of segments)			
	Model 1	**Model 2**	**Model 3**
Frequency (MHz)	300	600	900
Number of segments	7	22	51
Number of plates	128 (95.5 %)	128 (95.5 %)	128 (95.5 %)
Number of cylinders	6 (4.5 %)	6 (4.5 %)	6 (4.5 %)

Model Set 2 (increasing model size)			
	Model 4	**Model 5**	**Model 1**
Frequency (MHz)	300	300	300
Number of segments	7	7	7
Number of plates	64 (94.0 %)	84 (94.0 %)	128 (95.5 %)
Number of cylinders	4 (6.0 %)	5	6 (6.0 %) (4.5 %)

Model Set 3 (tertiary rays)	
	Model 6
Frequency (MHz)	300
Number of segments	7
Number of plates	20 (100.0 %)
Number of cylinders	0 (0.0 %)

perNEC compile configuration in the 2nd last line of the table indicates that the compiler optimisations (for Intel Pentium 4 processors [12]) are used and debugging is disabled.

Ray-Types For the tests 3 groups of ray-types are used which are listed in table C.2. The simple ray-types include direct rays (ie. line of sight) and single reflection and diffraction rays. The combined ray-types include the simple rays and all second order rays, ie. at most 2 arbitrary combinations of reflections and diffractions (using cylinders, end caps and plates). In a third group all tertiary rays are classed which consists of 3 arbitrary combinations of reflections and diffractions.

Test Models Six models divided into 3 sets are used to test the octree. In table C.3 the various characteristics of the models are listed. In the first set which consists of 3 models, the number of segments is varied but the number of components is kept constant. In the second set, the number of components is varied and the number of segments is kept constant. Model sets 1 and 2 are used to test both the simple and combined rays. For the tertiary rays model 6 is used. Model 1 is listed twice because it is used in experiment sets 1 and 2. The percentages next to the number of plates and cylinders are relative to the total number of components in the model.

The 6 models are used to determine the radiation patterns listed in table C.4. The top half

Table C.4: Radiation patterns. The column *degrees* gives the number of sample points taken in the azimuth or in the elevation planes.

Simple Ray Tests						
	θ			ϕ		
Plane	Start	End	Degrees	Start	End	Degrees
3D	0.0°	180.0°	181	0.0°	360.0°	361

Combined and Tertiary Ray Tests						
	θ			ϕ		
Plane	Start	End	Degrees	Start	End	Degrees
XY Plane	90.0°	90.0°	1	0.0°	360.0°	181
XZ Plane	0.0°	360.0°	181	0.0°	0.0°	1
YZ Plane	0.0°	360.0°	181	90.0°	90.0°	1

Table C.5: Overhead results for all model sets.

Model Set 1			
	Model 1	Model 2	Model 3
Construction Time (sec)	0.02	0.02	0.02
Memory (byte)	19737	19737	19737

Model Set 2			
	Model 4	Model 5	Model 1
Construction Time (sec)	0.01	0.01	0.02
Memory (byte)	10600	12986	19737

Model Set 3 (tertiary rays)	
	Model 6
Construction Time (sec)	0.01
Memory (byte)	2292

of the table lists the radiation pattern for simple rays. The radiation patterns used with all other ray groups are shown below. For each radiation pattern in the table, the plane, θ and ϕ ranges are given. For the simple rays, a full 3 dimensional radiation pattern is calculated. Three 2 dimensional radiation patterns are calculated for the other 2 ray groups.

C.3.2 Overhead

In this section the overhead results (in terms of memory and time) for the test models is presented and discussed. The octree described in section C.2, is constructed in a pre-processing stage and is stored in memory. The time to construct the octree and the memory usage are listed in table C.5 for all the test models. The table is divided into 3 sections; one for each model set.

Construction Time The construction time is the time required to build the octree. Construction time is only a function of the model size. The octree is in essence a 3D grid overlaid onto the model. Each component in the model must be allocated to a block (voxel) in that grid. The octree is built recursively using a divide and conquer approach. When a voxel is subdivided, only the components in that voxel are examined to determine in which child voxels the components should be placed. This reduces the number of components to be examined because only a subset of the total number of components are present in a (parent) voxel. The divide and conquer strategy leads to fast construction times.

Table C.5 shows that the construction time is negligible when compared with the overall run time (see the next section). For all models the construction time is under a second. The number of segments (ie. the model frequency) does not influence the construction time.

Memory Overhead The octree is a tree data structure. Each node in the tree has up to 8 children. A leaf node stores a list of pointers to the actual components (ie. the cylinders and plates making up the model). A leaf node only uses memory if it contains at least one component. Often a node will be subdivided into 8 leaf nodes and only one leaf is non-empty. For the empty leafs no memory is allocated. The size of the octree is only dependent on the number of components in the model and their layout.

Table C.6: Time results for model set 1 using simple rays.

	Model 1			Model 2			Model 3		
	Orig	**Opt**	×	**Orig**	**Opt**	×	**Orig**	**Opt**	×
Overhead	0.00	0.02		0.00	0.02		0.00	0.02	
Path Finding	683.29	684.22		1954.30	1959.15		4282.76	4305.14	
Shadow Tests	1168.21	146.73	7.96	1879.16	219.22	8.57	3741.91	433.55	8.63
EM	288.89	292.60		571.46	575.10		1161.57	1168.38	
Total Time	2140.39	1123.58	1.90	4404.92	2753.48	1.60	9186.25	5907.09	1.56

Table C.7: Time results for model set 2 using simple rays.

	Model 4			Model 5			Model 1		
	Orig	**Opt**	×	**Orig**	**Opt**	×	**Orig**	**Opt**	×
Overhead	0.00	0.01		0.00	0.01		0.00	0.02	
Path Finding	445.81	450.04		518.46	524.54		683.29	684.22	
Shadow Tests	392.47	57.67	6.81	604.69	68.41	8.84	1168.21	146.73	7.96
EM	162.49	165.41		206.23	206.29		288.89	292.60	
Total Time	1000.77	673.13	1.49	1329.38	799.25	1.66	2140.39	1123.58	1.90

The memory used for each model is listed in table C.5. On average each component requires 150 bytes storage space in the octree. The example models show a proportional relationship between model size and memory. Doubling the model size, doubles the memory used by the octree.

C.3.3 Results for simple rays

In this section results are presented for simple rays. When simple rays are used, the shadow time occupies a large portion of the runtime. Between 39.20 % and 54.60 % of run time is spent on the shadow tests. Using the octree both the shadow time and run time are significantly reduced. On average, the run time is a factor of 1.64 faster and the shadow tests are a factor of 8.16 faster.

The time results in tables C.6 and C.7 are gathered from model sets 1 and 2. The rows in each table contain the overhead time (construction time for the octrees), the path finding time, shadow test time, electromagnetic (EM) calculation time and the total run time. For each model, the times for the original SuperNEC and the optimised SuperNEC are listed. The speed-up factors for the shadow test and total times are listed in the third column (marked with ×).

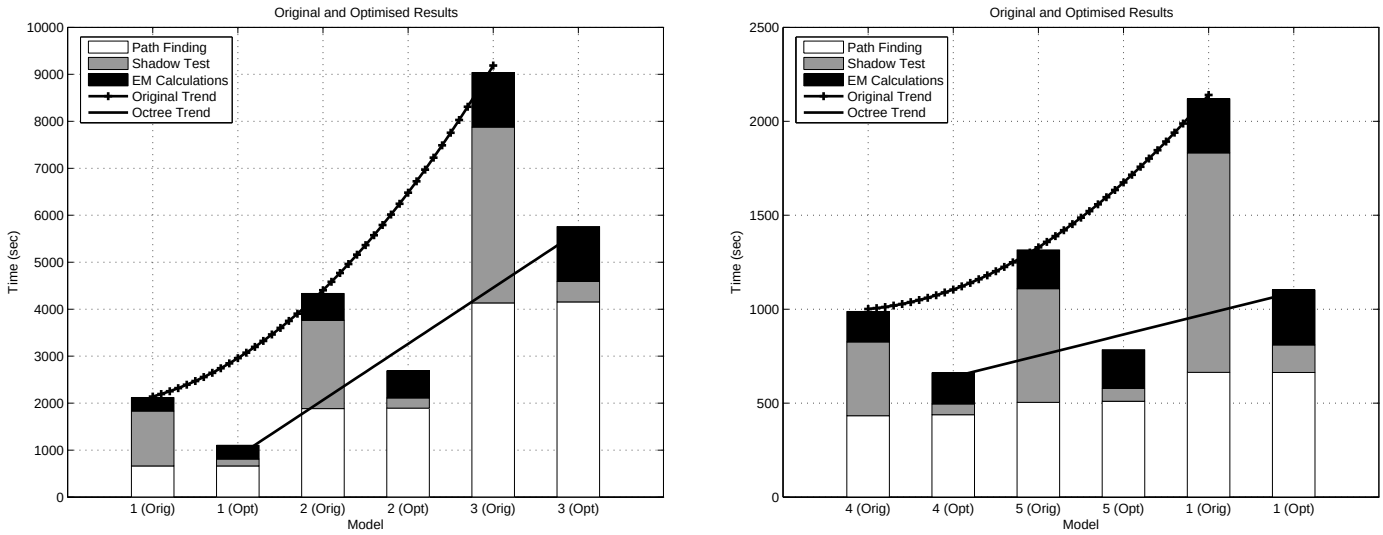
The data in tables C.6 and C.7 is plotted in figure C.5. A pair of bars show the run time for the original (orig) and optimised (opt) programs. Each bar shows the path finding time, shadow test time and the time for the electromagnetic (EM) calculations. The trend curves show that the increase in run time is exponential for the original program and linear for the optimised program.

The shadow test time is significantly reduced. The speed-up factor is in the range 6.81 to 8.63. On average the shadow test time for simple rays is 87 % less than the original time. The bar graphs in figure C.5(b) illustrate the significant reduction in the shadow time. Overall the run time is reduced by a factor of between 1.49 and 1.90. The run time is on average 40 % less than original path finding time.

The percentage of run time used for the different steps in SuperNEC is listed in tables C.8 and C.9 for model sets 1 and 2 respectively. Column 1 lists the different sections of SuperNEC. For each model, the percentages of the run time for the original and optimised

Table C.8: Percentage results for model set 1 using simple rays.

	Original	**Optimised**	**Original**	**Optimised**	**Original**	**Optimised**
Construction	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Path Finding	31.92%	60.90%	44.37%	71.15%	46.62%	72.88%
Shadow Tests	54.58%	13.06%	42.66%	7.96%	40.73%	7.34%
EM Calculations	13.50%	26.04%	12.97%	20.89%	12.64%	19.78%
Total Run Time	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%



(a) Increasing number of segments.

(b) Increasing model size.

Figure C.5: Run time for model sets 1 and 2 using simple rays.

Table C.9: Percentage results for model set 2 using simple rays.

	Original	**Optimised**	**Original**	**Optimised**	**Original**	**Optimised**
Construction	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Path Finding	44.55%	66.86%	39.00%	65.63%	31.92%	60.90%
Shadow Tests	39.22%	8.57%	45.49%	8.56%	54.58%	13.06%
EM Calculations	16.24%	24.57%	15.51%	25.81%	13.50%	26.04%
Total Run Time	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

SuperNEC are shown in the columns that follow.

On average the shadow tests use 9.10 % of run time using the optimisations. In the original SuperNEC the shadow tests occupied on average 45.45 % of the run time. After the optimisation, path finding time occupies a larger part of the run time. Previously path finding and the shadow tests both took up an equally important share of the run time. Using the octree, path finding takes up the majority of the run time.

Table C.10: Number of intersection tests per ray for models in set 1 using simple rays for the original (orig) and optimised (opt) programs. Column × is the reduction from the original to the optimised program.

Intersections per ray	**Orig**	**Opt**	×	**Orig**	**Opt**	×	**Orig**	**Opt**	×
Average	114.71	5.64	20.35	107.70	5.29	20.35	108.25	5.68	19.07
Maximum	134	42	3.19	134	42	3.19	134	41	3.27
Minimum	1	0		1	0	0	1	0	0

In tables C.10 and C.11 the average, maximum and minimum number of intersections per ray for all the models are listed. The maximum number of intersections per ray for the original program for all models equals the total number of components. If a ray is shadowed, then originally at least 1 intersection test is conducted. The average number of intersections per ray is then expected to be close to the maximum number of intersection tests. This is confirmed by the experimental results.

Using the octree, the maximum number of intersections per ray is significantly reduced. During the experiments the octree reduced the number of intersections per ray by a factor between 15.03 and 23.19. For a ray that is not shadowed, fewer than the maximum number of components need to be inspected. The maximum number of ray intersections is reduced by between 2.41 and 3.27 when compared with the original program. The minimum number of intersections per ray is reduced to 0.

In model set 1 the number of segments is varied. More rays are traced as the number of seg-

Table C.11: Number of intersection tests per ray for models in set 2 using simple rays for the original (orig) and optimised (opt) programs. Column × is the reduction from the original to the optimised program.

Intersections per ray	Orig	Opt	×	Orig	Opt	×	Orig	Opt	×
Average	62.79	4.18	15.03	82.19	3.54	23.19	114.71	5.64	20.35
Maximum	68	27	2.52	89	37	2.41	134	42	3.19
Minimum	1	0		1	0	0	1	0	0

Table C.12: Time results for model set 1 using combined rays.

	Model 1			Model 2			Model 3		
	Orig	**Opt**	×	**Orig**	**Opt**	×	**Orig**	**Opt**	×
Overhead	0.00	0.02		0.00	0.02		0.00	0.02	
Path Finding	9089.66	9090.50		35700.87	35711.83		93960.69	93931.16	
Shadow Tests	310.69	62.71	4.95	820.51	150.59	5.45	2219.38	385.21	5.76
EM	124.49	122.82		374.11	380.30		155.86	153.48	
Total Time	9524.84	9276.05	1.03	36895.49	36242.74	1.02	96335.93	94469.86	1.02

ments increases. The run time of the original brute force algorithm increases exponentially as the number of rays increase. The scalability of the octree means the speed-up will be very large. The speed-up increases from 7.96 to 8.63. However as the number of rays increase, path finding time increases (see table C.6). So the overall speed-up tends to decrease with increasing number of segments.

In model set 2 the number of components is varied. Increasing the number of components, increases the number of possible ray paths. Path finding time increases and thus shadow test time also increases because more rays are traced. In the original program, as model size increases the shadow test time as a percentage of the total run time increases and path finding time as a percentage of the total run time decreases. The importance of the shadow tests in this case means that the optimisation speed-up (for the shadow tests and for the total run time) tends to increase with increasing model size.

C.3.4 Results for combined rays

In this section results for the shadow test optimisations are presented for the combined rays. When using second order rays, the path finding time dominates over the shadow test time. The run time is not appreciably increased (the speed-up is approximately unity for all models). However, the shadow tests are a factor of 5 faster on average (which is similar to the results for simple rays).

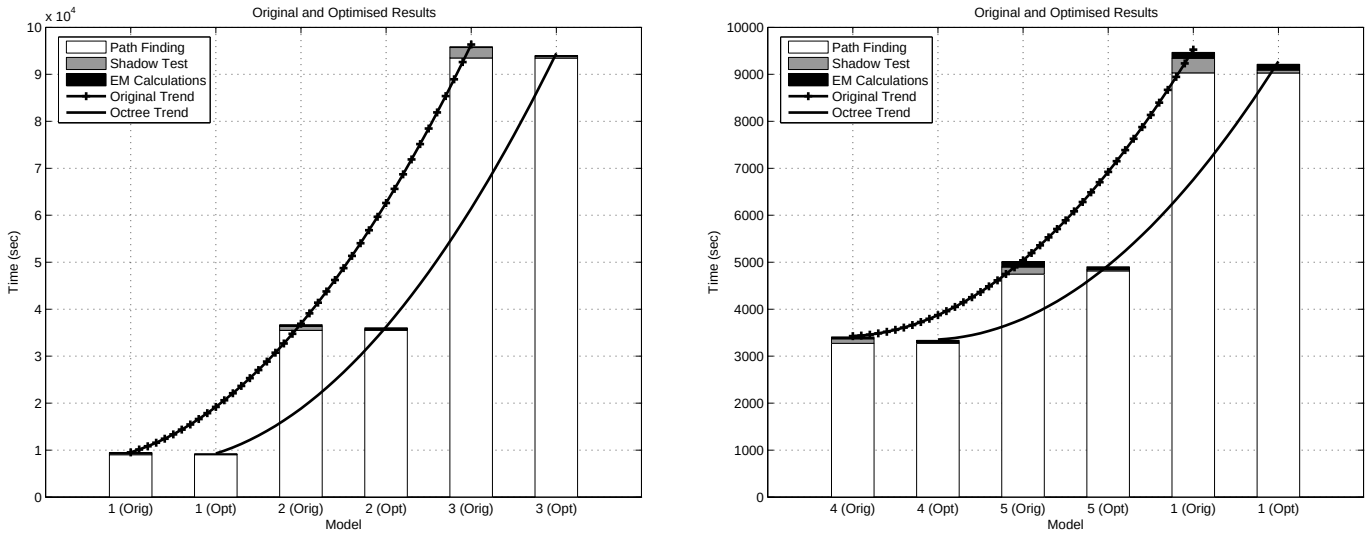
Table C.13: Time results for model set 2 using combined rays.

	Model 4			Model 5			Model 1		
	Orig	**Opt**	×	**Orig**	**Opt**	×	**Orig**	**Opt**	×
Overhead	0.00	0.01		0.00	0.01		0.00	0.02	
Path Finding	3289.75	3291.80		4776.15	4843.34		9089.66	9090.50	
Shadow Tests	98.27	23.83	4.12	149.69	28.32	5.29	310.69	62.71	4.95
EM	38.58	38.86		113.68	59.48		124.49	122.82	
Total Time	3426.59	3354.50	1.02	5039.52	4931.16	1.02	9524.84	9276.05	1.03

Tables C.12 and C.13 lists the timing results for all 5 models. The speed-up factors (column ×) are shown for the shadow test and total run times. In this case the path finding time is exceptionally high. Using the octree the shadow tests are a factor of between 4.12 and 5.45 faster. However most of the run time is spent on the path finding and the overall speed-up is approximately unity for all models.

Figure C.6 shows the results from tables C.12 and C.13. The structure of the graphs is the same as in the previous section for simple rays. The trend curves indicate that using the shadow tests with second order rays does not reduce the run time. In both figures the curves run almost parallel and increase exponentially.

Tables C.14 and C.15 show the percentage of total run time of each section in SuperNEC. Path finding takes up more than 95 % of the total time in all models before and after the



(a) Increasing number of segments.

(b) Increasing model size.

Figure C.6: Run time for model sets 1 and 2 using combined rays.

Table C.14: Percentage results for model set 1 using all rays.

	Original	**Optimised**	**Original**	**Optimised**	**Original**	**Optimised**
Construction	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Path Finding	95.43%	98.00%	96.76%	98.54%	97.53%	99.43%
Shadow Tests	3.26%	0.68%	2.22%	0.42%	2.30%	0.41%
EM Calculations	1.31%	1.32%	1.01%	1.05%	0.16%	0.16%
Total Run Time	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

shadow test optimisations. On average the shadow tests take up only 2.80 % of the total time.

Tables C.16 and C.17 lists the average, maximum and minimum number of intersections per ray. As discussed previously in section C.3.3 for the case of simple rays, these results show how the octree focuses the intersection tests on those components which are situated in the vicinity of a ray. On average there are between 6.82 and 11.26 fewer intersection tests per ray. The maximum number of ray intersections is reduced by between 2.28 and 3.35. For certain rays, the octree is able to avoid an intersection test. The minimum number of intersection tests per ray using the octree is 0.

In both model sets with varying number of segments and varying number of components, path finding time dominates. The octree decreases the shadow test time significantly but path finding time is so high that the overall speed-up using only the octree is unity for all models.

C.3.5 Results for tertiary rays

In this section the results are presented for the tertiary rays. As with the combined rays, path finding time takes up the majority of the run time. Shadow test time is negligible. The overall run time remains unchanged using the octree. The shadow test time is reduced by a factor of 1.80. The reason for such a low figure (compared with the results from the

Table C.15: Percentage results for model set 2 using all rays.

	Original	**Optimised**	**Original**	**Optimised**	**Original**	**Optimised**
Construction	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Path Finding	96.01%	98.13%	94.77%	98.22%	95.43%	98.00%
Shadow Tests	2.87%	0.71%	2.97%	0.57%	3.26%	0.68%
EM Calculations	1.13%	1.16%	2.26%	1.21%	1.31%	1.32%
Total Run Time	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

Table C.16: Number of intersection tests per ray for models in set 1 using simple and second order rays for the original (orig) and optimised (opt) programs. Column × is the reduction from the original to the optimised program.

Intersections per ray	**Orig**	**Opt**	×	**Orig**	**Opt**	×	**Orig**	**Opt**	×
Average	86.22	8.20	10.52	77.01	6.84	11.26	77.69	7.05	11.02
Maximum	134	41	3.27	134	40	3.35	134	40	3.35
Minimum	1	0		1	0	0	1	0	0

Table C.17: Number of intersection tests per ray for models in set 2 using simple and second order rays for the original (orig) and optimised (opt) programs. Column × is the reduction from the original to the optimised program.

Intersections per ray	**Orig**	**Opt**	×	**Orig**	**Opt**	×	**Orig**	**Opt**	×
Average	47.12	6.91	6.82	62.44	5.87	10.63	86.22	8.20	10.52
Maximum	68	26	2.62	89	39	2.28	134	41	3.27
Minimum	1	0		1	0	0	1	0	0

previous section) is that so few tertiary rays are found.

Table C.18 shows the time used by each section in SuperNEC. The shadow test time amounts to only 80 seconds in the original SuperNEC. The path finding requires just under 3 hours which is more than 100 times the shadow test time.

Table C.19 gives a breakdown of the run times in percentages. 97 % of the run time is used to find ray paths.

The average, maximum and minimum number of intersections per ray is listed in table C.20. These results are similar to the results for simple and combined rays. In the original program almost all the plates need to be inspected to determine if a ray intersects a plate. However, in the optimised version approximately 5 plates are inspected to determine if a ray intersects a plate.

The shadow optimisations significantly reduce the shadow test time. On a global scale the shadow optimisations do not optimise the run time of tertiary rays.

C.4 Scalability of Optimised Shadow Tests

In this section results are presented which compare the original and optimised shadow tests, to determine the scalability with model size and number of segments.

The run time of the original shadow algorithm increases exponentially with model size and linearly with number of segments [1]. Two sets of experiments are run to determine the how the shadow time varies with model size and number of segments using the octree. The time to access the octree is also investigated. In the first experiment the model size is varied and the number of segments kept constant. Next, the number of segments is varied and the model size kept constant. Eight models are used in each experiment.

An overview of the experiments is given in section C.4.1. In section C.4.2 the results of the experiments are presented.

Table C.18: Time results for model set 3 using tertiary rays.

	Model 6		
	Original (sec)	**Optimised** (sec)	**Speed-up**
Overhead	0.00	0.01	
Path Finding	10525.80	10523.90	
Shadow Tests	81.47	45.14	1.80
EM	194.51	195.70	
Total Time	10801.78	10764.74	1.00

Table C.19: Percentage results for model set 3 using tertiary rays.

	Model 6	
	Original	**Optimised**
Overhead	0.00 %	0.00 %
Path Finding	97.45 %	97.76 %
Shadow Tests	0.75 %	0.42 %
EM	1.80 %	1.82 %
Total Time	100.00 %	100.00 %

Table C.20: Number of intersection tests per ray for model set 3.

Intersections per ray	**Original**	**Optimised**	Factor
Average	17.27	4.18	4.14
Maximum	20	16	1.25
Minimum	1	0	

C.4.1 Overview of the Tests

Hardware and software for the tests. The hardware and software is described in table C.1 (in section C.3.1).

Ray types for the tests. For simplicity only direct, single diffraction and single reflection rays are used. The shadow test algorithm processes rays and is not aware of the ray types. There is no difference between a direct ray and a more complex ray, eg. double diffraction.

Test models. Table C.21 lists the 16 models that are used to examine the scalability of the octree. All models only contain plates. This was done for simplicity. The models in

Table C.21: Models for scalability tests.

(a) Increasing model size.

Frequency: 1000 MHz, Number of segments: 34								
Model	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Number of plates	3	6	14	25	35	49	98	147

(b) Increasing number of segments.

Number of plates	98							
Model	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Frequency (MHz)	250	500	750	1000	1250	1500	1750	2000
Segments	10	18	26	34	42	54	62	70

table C.21(a) have a variable number of components and constant number of segments. In table C.21(b) the models have the same size but variable number of segments.

Radiation Patterns. Details of the radiation patterns used during the tests are given in table C.22. For the tests, full 3D radiation patterns are calculated.

C.4.2 Results

Increasing model size

The relationship between shadow time and model size is investigated using the 8 models described in table C.21(a). The shadow time for the original and optimised programs while calculating the radiation patterns is given in C.23. The table also lists the average shadow time per plate (time/plate) and the average number of intersections per ray.

The average processing time per plate and the average number of intersections per ray is plotted in figure C.7(a) and C.7(b) respectively. Model size increases along the x-axis. The trend curve shows the exponential nature of the original program. A second trend curve shows the linear nature of the octree program.

Table C.22: Radiation patterns for scalability tests. The column *degrees* gives the number of sample points taken in the azimuth or in the elevation planes.

	θ			ϕ		
Plane	Start	End	Degrees	Start	End	Degree Degrees
3D	0.0°	180.0°	181	0.0°	360.0°	361

Table C.23: Shadow test time and number of intersections per ray for the original and optimised programs for increasing model size.

Original								
Time (sec)	6.91	13.44	46.84	62.92	271.85	402.15	1177.13	2969.20
Time/plate (sec/plate)	2.31	2.24	5.86	4.50	7.77	8.21	12.01	20.20
Intersections per ray	2.33	4.36	6.36	11.77	28.22	29.06	58.92	96.57
Optimised								
Time (sec)	6.87	12.22	39.39	31.96	99.61	121.74	219.73	374.32
Time/plate (sec/plate)	2.29	2.04	4.92	2.28	2.85	2.49	2.24	2.55
Intersections per ray	0.54	0.93	0.95	1.46	4.56	3.47	4.68	4.44

For the original shadow tests the average number of intersections per ray is high. On average more than 50 % of the components in a model are examined. In the worst case, when the ray does not intersect a component the original shadow tests examines all components. For the octree, the ray is only intersected with components that lie in its vicinity. The average is therefore very low for the octree.

Table C.24: Shadow time decomposition. Octree access time and percent of the total shadow time is given in the first 2 rows. The time and percentages for the intersection tests are given below that. The final row is the total shadow time.

Increasing model size								
Access (sec)	4.93	8.11	27.63	19.84	47.61	58.60	95.47	164.30
Access %	**71.74 %**	**66.37 %**	**70.13 %**	**62.08 %**	**47.79 %**	**48.14 %**	**43.45 %**	**43.89 %**
Tests (sec)	1.94	4.11	11.77	12.12	52.00	63.14	124.25	210.03
Tests %	28.26 %	33.63 %	29.87 %	37.92 %	52.21 %	51.86 %	56.55 %	56.11 %
Total (sec)	6.87	12.22	39.39	31.96	99.61	121.74	219.73	374.32

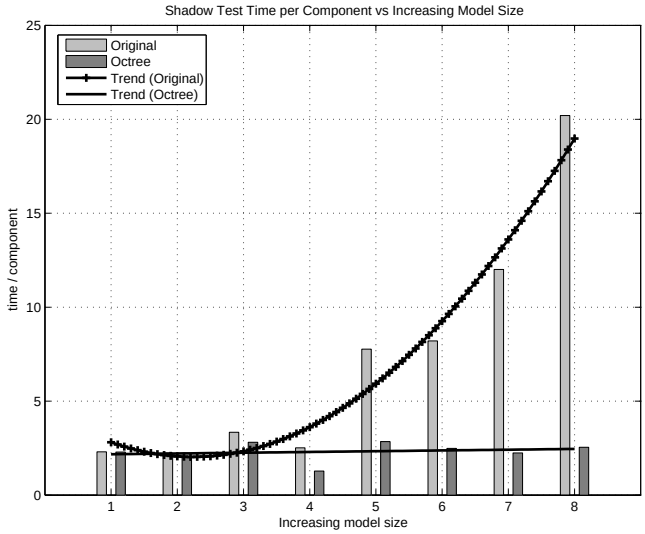
Table C.24 lists the time to access the octree. Before the intersection tests the voxels intersected by the ray are located (explained in section C.2.2). The time to find the intersected voxels is the access time. For the 8 models the access time and the percentage of the total shadow time required to access the octree is given in table C.24. For comparative purposes, the time for the intersection tests and the relevant percentages are given as well as the total shadow time.

As the model size increases the access time as a percentage of the shadow time decreases. This is to be expected because for a small model once the voxels intersected by the ray are located, only few intersection tests are needed to determine if the ray is shadowed. For a larger model, once the voxels are found many tests are required to determine which component is blocking the ray. Thus the overhead is higher for a smaller model than for a larger model.

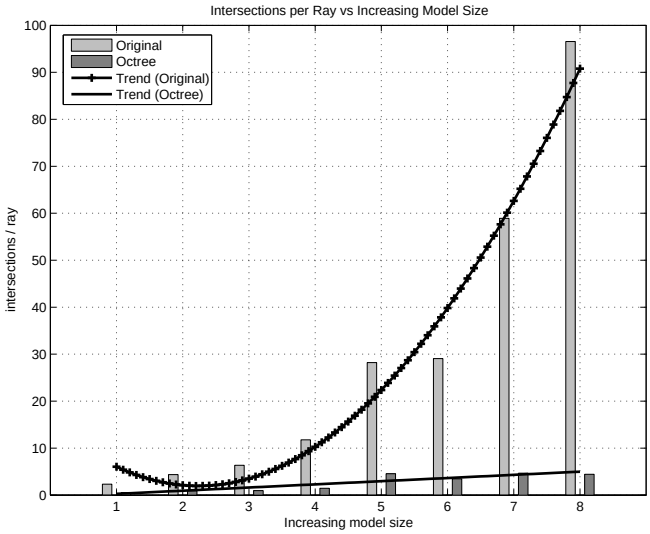
The results show that, using the octree, the shadow tests scale well with model size. Using the octree the relationship between model size and the shadow time is linear. The overhead to access the data in the octree is above 50 % for small models (< 25 components) and less the 50 % as the model size increases. This is acceptable considering the speed-up and the linear nature of the optimised shadow tests.

Increasing number of segments

The relationship between the shadow time and the number of segments is investigated using the models listed in table C.21(b). For each model a 3 dimensional radiation pattern is



(a) Shadow time per component.



(b) Average intersections per ray.

Figure C.7: Shadow time and number of intersections per ray for models with increasing size.

Table C.25: Shadow test time and number of intersections per ray for the original and optimised programs for increasing number of segments.

Original								
Time (sec)	340.81	622.32	900.94	1179.20	1465.70	1879.24	2159.41	2429.27
Intersections per ray	58.97	58.96	58.93	58.92	58.92	58.91	58.78	58.66
Optimised								
Time (sec)	64.54	116.66	166.368	220.56	270.53	350.98	398.43	450.55
Intersections per ray	4.59	4.64	4.40	4.68	4.62	4.60	4.62	4.63

calculated. The shadow time for the original and optimised programs are listed in table C.25. The shadow times are plotted in figure C.8. The values on the x-axis correspond to the 8 models with increasing number of segments.

For each extra segment more rays are launched and more shadow tests are conducted. Keeping the model size constants means that for each ray there are on average the same number of intersection tests. Consequently an increase in the number of segments leads to a proportional increase in the shadow time. In figure C.8, the trend curves for the original and the octree algorithms indicate the linear nature between number of segments and shadow time.

Using the octree fewer intersection tests are done per ray because the octree only tests those components that are within the vicinity of the ray. Thus the increase in octree shadow time is smaller as the number of segments increases when compared with the original algorithm. Table C.25 lists the average number of intersection tests for each ray for the original and optimised programs. In both cases the values are fairly constant but the octree does fewer intersection tests for each ray for the reasons just mentioned.

Table C.26: Shadow time decomposition. Octree access time and percent of the total shadow time is given in the first 2 rows. The time and percentages for the intersection tests are given below that. The final row is the total shadow time.

	Increasing number of segments							
Access (sec)	28.38	53.20	73.90	96.18	119.77	157.31	177.00	196.16
Access %	**43.98 %**	**45.60 %**	**44.42 %**	**43.61 %**	**44.27 %**	**44.82 %**	**44.42 %**	**43.54 %**
Tests (sec)	36.16	63.46	92.47	124.39	150.76	193.68	221.43	254.38
Tests %	56.02 %	54.40 %	55.58 %	56.39 %	55.73 %	55.18 %	55.58 %	56.46 %
Total (sec)	64.54	116.66	166.368	220.56	270.53	350.98	398.43	450.55

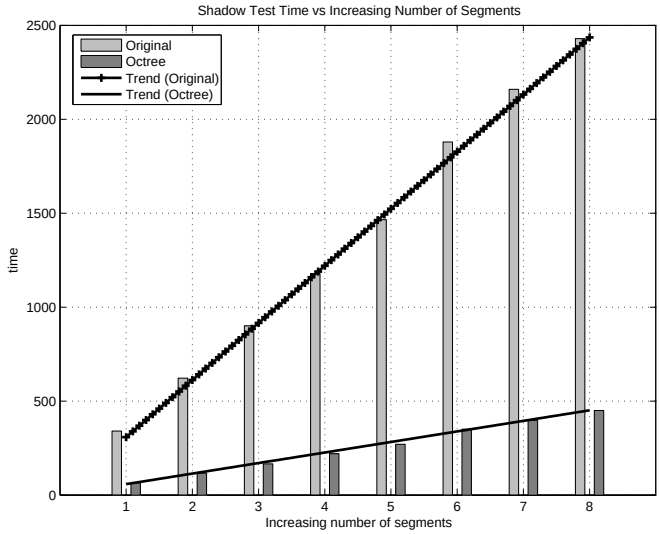


Figure C.8: Shadow time for models with increasing number of segments.

The octree access time is given in table C.26. The access time expressed as a percentage of the total shadow time is constant (approximately 44 %). The number of components is fixed and the octree is the same in each model. Although a different number of rays are traced for each model the average number of voxels that are accessed for each ray remains constant. The access time increases as the number of segments increases but the access time expressed as a percentage of the shadow time remains constant.

C.4.3 Summary of Scalability Tests

The run time of the optimised shadow tests are independent of model size and the number of segments. The optimised shadow tests scale well. The run time of the original algorithm increases exponentially with model size and the optimised shadow tests increase linearly with model size. In the original program for each ray more than half the components are examined to determine if the ray is shadowed. The optimised program examines on average less than 5 % of the components in a model. The octree access time takes up a significant part of the shadow time. This is acceptable given the speed-up and the linear nature of the optimisations.

C.5 Conclusion

The report shows how an octree is used in SuperNEC-UTD to optimise the shadow tests. For simple rays the octree significantly reduces the overall run time. When higher order rays are used the run time cannot be reduced using only octree optimisations. The optimised shadow tests scale with increasing model size and increasing number of segments.

The time to determine far field radiation patterns using only simple rays decreases by a factor of 1.70. The shadow test time decreases by a factor of 8. The shadow test time for higher order rays is reduced by a factor of 5. Overall the run-time when higher order rays are used is not reduced because the time for path finding is much larger than the shadow time.

References

[1] R. Hartleb, "Optimised ray tracing for the SuperNEC implementation of the uniform theory of diffraction," MSc(Eng) Dissertation, University of the Witwatersrand, Johannesburg, 2006, Appendix A: Analysis of the run time of SuperNEC-UTD: a MoM electromagnetic simulation package.

[2] H. Samet, *The design and analysis of spatial data structures.* Addison-Wesley, 1990.

[3] ——, "The quadtree and related hierarchical data structures," *ACM Computing Survey*, vol. 16, no. 2, pp. 187–260, June 1984.

[4] A. S. Glassner, Ed., *An Introduction to Ray Tracing.* Academic Press, 1989.

[5] A. Glassner, "Space subdivision for fast ray tracing," *IEEE Computer Graphics and Applications*, vol. 4, no. 10, pp. 15–22, Oct. 1984.

[6] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes, *Computer Graphics: Principles and Practice.* Addison-Wesley, 1990.

[7] A. Watt and F. Policarpo, *The Computer Image.* Addison-Wesley, 1998.

[8] A. Fujimoto, T. Tanaka, and K. Iwata, "Arts: Accelerated ray tracing system," *IEEE Computer Graphics and Applications*, vol. 6, no. 4, Apr. 1986.

[9] I. Carlbom, I. Chakravarty, and D. A. Vanderschel, "A hierarchical data structure for representing the spatial decomposition of 3D objects," *IEEE Computer Graphics & Applications*, vol. 5, no. 4, pp. 24–31, 1985.

[10] K. Subramanian and D. Fussel, "A search structure based on k-d trees for efficient ray tracing," 1992, last accessed 27 January 2006. [Online]. Available: citeseer.ist.psu.edu/subramanian92search.html

[11] Y. D. Liang and B. A. Barsky, "A new concept and method for line clipping," *ACM Transactions on Graphics*, vol. 3, no. 1, pp. 1–22, Jan. 1984.

[12] Intel pentium 4 processor. Intel Corporation. Last accessed 3 April 2006. [Online]. Available: http://www.intel.com/products/processor/pentium4/

Appendix D

Collective Results of Geometric Optimisations used in SuperNEC-UTD

Abstract

Results of tests of the geometric optimisations used collectively to decrease the run time to calculate radiation patterns in SuperNEC-UTD are presented. Four geometric optimisations are used: back face culling, reflection zones, diffraction zones and octree. The tests look at the performance of only simple rays (line of sight, single reflections and diffractions), of all rays up to second order and of all rays up to third order. Using the optimisations collectively the run time is reduced by half on average. The optimisation techniques introduces errors into the radiation patterns. In general the errors are localised to small ranges in the azimuth and elevation planes. The mean absolute errors are approximately 0.02 dB for simple rays and when higher order rays are used approximately 0.17 dB.

D.1 Introduction

In this report back face culling (BFC), reflection zone (RZ) and diffraction zone (DZ) optimisations for path finding and the octree optimisations for the shadow tests are examined collectively. All ray-types avaliable in SuperNEC-UTD are enabled.

The path finding optimisations reduce the run time by determining in advance the spatial regions which are illuminated by a particular ray-type. The shadow tests are optimised using an octree which is a non-uniform subdivision of the volume occupied by the model. An octree allows quick access to the components which are located close to a ray.

The report is divided into 3 sections. First, in section D.2 details on the various tests are given. Next, in section D.3 the results of tests that only use simple rays are presented. Then in section D.4 the results for tests where both simple and second orders rays are used are discussed. Finally in section D.5 the results of tests for all ray types including tertiary rays are presented.

D.2 Overview of the Tests

This section provides information about the tests which are used to examine the optimisations.

Hardware and software for the tests. All tests in sections B.3 to B.5 are run using the hardware and software described in table D.1. Table B.4 is for the most part self-explanatory.

Table D.1: Hardware and software used during the tests.

Hardware	
Processor	Intel(R) Pentium(R) 4 CPU [1]
Processor Frequency	3.80 GHz
Random Access Memory (RAM)	1.00 GB
Software	
Operating System (OS)	Microsoft Windows XP Professional
OS Version	Version 2002, Service Pack 2
SuperNEC Version	2.9
SuperNEC Compile Configuration	WIN32 Release
Compiler	Microsoft Visual C++ 6.0

SuperNEC compile configuration in the 2nd last line of the table indicates that the compiler optimisations are used and debugging is disabled.

Ray-Types The ray-types used during the tests are listed in table D.2. The rays are divided into simple, second order and tertiary rays. For each group the rays that interact with plates, cylinders, cylinder end caps and both plates and cylinders are given. Rays which are optimised are highlighted. The notation for the ray-types is explained [2].

The tests are structured according to the ray groups in table D.2. In one experiment only simple rays are used to calculate radiation patterns. In the next experiment both simple and second order rays are combined. Finally all rays (simple, second order and tertiary rays) are used to calculate radiation patterns.

Models Table D.3(a) lists the 3 models which are used to test simple and second order rays. In table D.3(b) the model used to test tertiary rays is given. The geometry of the models was kept as general as possible. Each path finding optimisations requires specific geometric structures. Wedges are used with diffraction zones. In the row labelled wedges the number of edges is given in brackets. Polyhedrons are exploited by back face culling and the diffraction zones. The number of plates used for the polyhedrons is given in brackets. Each model has single (not connected) plates. All plates are optimised using reflection zones.

Table D.2: Ray-types for the 3 experiment sets.

Simple Rays (first order and direct rays)	
Plate	
Optimised	RP, DE
Cylinder	
Un-optimised	RC, DC
End-Cap	
Un-optimised	RL, DL
Second Order Rays	
Plate	
Optimised	RP-RP, RP-DE, DE-RP, DE-DE
Cylinder	
Un-optimised	RC-RC, RC-DC, DC-RC, DC-DC
End-Cap	
Un-optimised	DL-DL
Plate and Cylinder Combined	
Optimised	RP-RC,RP-DC, DE-RC,DE-DC, RC-DE,DC-DE
Un-optimised	RC-RP,DC-RP
Tertiary Rays	
Plate	
Optimised	RP-RP-RP, DE-RP-RP, DE-RP-DE, DE-DE-RP, DE-DE-DE

Table D.3: Test models for the experiments.

(a) Test models for simple rays and combined rays.

Model	**1**	**2**	**3**
Frequency (MHz)	300	300	300
Segments	7	7	7
Plates (total)	80	117	157
Single plates	3	3	3
Wedges	151 (267)	213 (378)	308 (546)
Polyhedrons	2 (38)	3(58)	5 (82)
Cylinders	2	8	12

(b) Test model for all rays.

Model	**1**
Frequency (MHz)	300
Segments	7
Plates (total)	20
Single plates	4
Wedges	24 (40)
Polyhedrons	4 (16)
Cylinders	2

Radiation Patterns For each model the radiation patterns in table D.4 are calculated. A 3 dimensional radiation pattern is calculated when only simple rays are used. When both simple and second order rays are used three 2 dimensional radiation patterns are calculated. When all rays are enabled three 2 dimensional radiation patterns are calculated. For second and higher order rays only a 2D radiation pattern is calculated to keep the run time reasonable.

Table D.4: Radiation patterns for test models. The column *degrees* gives the number of sample points taken in the azimuth or in the elevation planes.

Simple Rays						
	θ			ϕ		
Plane	Start	End	Degrees	Start	End	Degrees
3D	0.0°	180.0°	181	0.0°	360.0°	361
Combined Rays (simple and second order rays)						
XY Plane	90.0°	90.0°	1	0.0°	360.0°	181
XZ Plane	0.0°	360.0°	181	0.0°	0.0°	1
YZ Plane	0.0°	360.0°	181	90.0°	90.0°	1
All Rays (simple, second order and tertiary rays)						
XY Plane	90.0°	90.0°	1	0.0°	360.0°	181
XZ Plane	0.0°	360.0°	181	0.0°	0.0°	1
YZ Plane	0.0°	360.0°	181	90.0°	90.0°	1

Table D.5: Run times for models 1 to 3 using simple rays for the original (orig) program, using back face culling (BFC), using reflection zones (RZ), using diffraction zones (DZ), using the octree and using all optimisations. All values in seconds.

Model 1	**Orig**	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Path Finding	259.88	232.78	306.00	228.29	260.06	247.39
Shadow Time	387.17	373.89	385.64	337.46	88.79	66.83
EM	221.31	198.35	217.22	205.62	213.75	189.81
Overall Run Time	**868.36**	**805.03**	**909.02**	**771.45**	**562.61**	**504.25**
Model 2	**Orig**	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Path Finding	703.68	660.62	770.17	656.60	677.36	690.41
Shadow Time	1064.55	1049.29	1073.81	981.36	170.04	146.75
EM	332.71	312.60	330.35	329.98	352.02	298.99
Overall Run Time	**2100.94**	**2022.53**	**2174.59**	**1967.94**	**1199.44**	**1136.49**
Model 3	**Orig**	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Path Finding	1032.99	976.14	1127.62	969.38	1001.64	1013.48
Shadow Time	2108.75	2090.68	2119.25	1976.14	254.64	224.97
EM	426.38	400.10	423.56	427.27	455.09	385.32
Overall Run Time	**3568.11**	**3466.95**	**3670.83**	**3373.03**	**1711.38**	**1624.28**

D.3 Simple Rays

D.3.1 Ray Tracing

Table D.5 lists the run times for each model. Vertically the table lists the time for path finding, shadow test, electromagnetic calculations and the overall time. Horizontally results are for the original program, the 4 individual techniques and the combination of the techniques.

In table D.6 the speed-up factors for path finding, shadow tests and the total run time are given. First the path finding results are shown for each ray individually and then the total path finding speed-up is given. Next the shadow tests results are given. In the final row the total speed-up is given. Results are given for each optimisation and in the last column for the combination of the optimisations[1].

Examining the total path finding time and speed-up, shows that when using simple rays, the path finding optimisations are not effective. The search space is too small which means that scalability does not pose a problem. The original brute force algorithm for path finding uses approximately the same amount of time as the 3 path finding techniques.

The shadow test time is significantly reduced by factors between 4 and 9 using the octree. Back face culling and diffraction zones are able to determine in advance if a component or far field point is visible from a segment or an edge. The 2 techniques reduce the number of rays passed to the shadow algorithm and therefore contribute to the reduction in the shadow test time. Overall the shadow tests are reduced by a factor of between 6 and 10. The high reduction factor for the shadow tests leads to an overall reduction in the run time by a factor of between 1.72 and 2.20. On average this is a 50 % reduction in the original run time.

In table D.7 the search space reductions are shown. On the left of the table the ray-types are listed. The top row lists the optimisation techniques. The row labelled *total* is the overall reduction. The search space results show essentially the same results as the previous tables. The path finding optimisations significantly reduce the search space for RP and DE rays. However the search space for cylinder rays is unchanged and the optimisations are not effective.

Shadow tests results are given in table D.8. Comparing the original with the octree optimised program, for each ray far fewer intersection tests are run on average to determine if a ray is blocked. The reduction row shows that using the octree the average number of intersections per ray is reduced by a factor between 10 and 23. BFC and DZ also reduce the number of intersection tests. The octree is aware of the location of the components and can sometimes determine without running any intersection tests that a ray is not blocked. The minimum number of intersection tests for the octree is therefore 0. For the octree the maximum

[1]The speed-up results in the octree column for path finding should all be 1. Due to background processes in the operating system the run times vary slightly which leads to speed-up factors different from 1.

Table D.6: Speed-up factors for models 1 to 3 using simple rays for back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), the octree and for all optimisations.

Model 1	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Path Finding RP	1.35	144.05	0.95	1.02	203.14
DE	1.32	1.02	1.70	0.97	2.03
RC	0.98	1.01	1.02	1.00	0.98
DC	1.02	0.99	1.02	0.99	0.97
REC	0.99	1.02	1.07	0.85	0.98
DEC	1.01	0.99	1.01	1.02	1.01
Total	1.12	0.85	1.14	1.00	1.05
Shadow Time	1.04	1.00	1.15	4.73	6.26
Overall Run Time	**1.08**	**0.96**	**1.13**	**1.54**	**1.72**

Model 2	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Path Finding RP	1.50	214.41	1.09	1.01	216.35
DE	1.34	1.01	1.72	1.00	2.10
RC	0.99	1.01	1.00	0.99	0.99
DC	1.01	1.00	0.99	1.01	0.96
REC	1.11	0.99	1.04	0.98	1.03
DEC	1.00	1.00	1.00	1.01	0.99
Total	1.07	0.91	1.07	1.04	1.02
Shadow Time	1.01	0.99	1.08	6.57	7.64
Overall Run Time	**1.04**	**0.97**	**1.07**	**1.75**	**1.85**

Model 3	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Path Finding RP	1.38	494.81	0.96	1.00	283.39
DE	1.28	1.01	1.69	0.99	1.90
RC	1.00	0.99	0.99	1.00	1.01
DC	1.01	1.01	0.98	0.99	1.01
REC	1.01	1.00	1.05	0.99	0.95
DEC	1.01	1.01	1.01	1.00	1.01
Total	1.06	0.92	1.07	1.03	1.02
Shadow Time	1.01	1.00	1.07	8.79	9.93
Overall Run Time	**1.03**	**0.97**	**1.06**	**2.08**	**2.20**

number of intersections is less than half the number of components in the model.

The reduction in the number of intersection tests increases when moving from model 1 to 3, ie. when the model size increases. As the model size increases the shadow time of the original program increases quadratically whereas for the optimised program the increase is linear and thus the speed-up increases.

D.3.2 Errors

Table D.9 lists the errors for simple rays[2]. Results are only specified for the techniques which cause errors. The mean absolute error (m_{ae}) is given. These values are for the full 3D radiation pattern. BFC does not produce any errors in these models. The mean error for DZ and when all optimisations are used is approximately 0.02 dB.

In figure D.1 the radiation patterns of the original and optimised programs for the ϕ-cuts with the highest mean absolute error are plotted. The error is localised to a small region of the plot (between $\theta = 120°$ and $\theta = 140°$).

[2]The error measure is discussed in [3]

Table D.7: Search space reduction for models 1 to 3 using simple rays for back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), octree and for all optimisations.

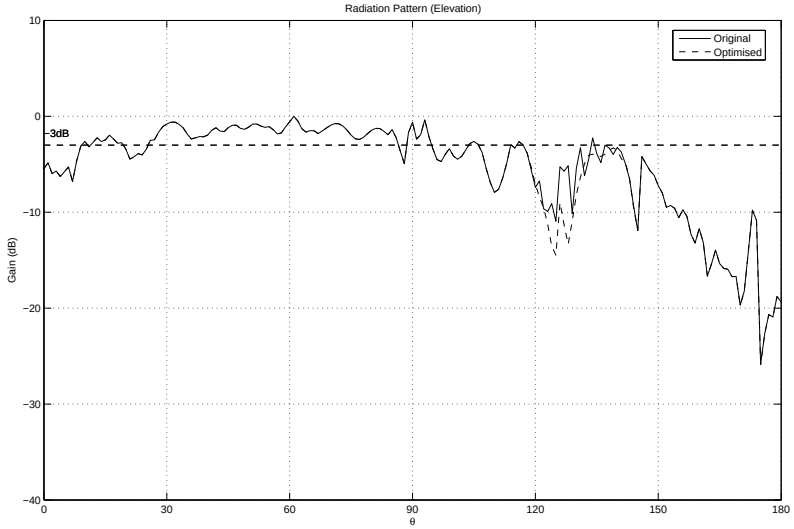
Model 1	**BFC**	**RZ**	**DZ**	**Octree**	**All**
RP	1.42	358.55	1.00	1.00	502.31
DE	1.25	1.00	1.52	1.00	1.75
RC	1.00	1.00	1.00	1.00	1.00
DC	1.00	1.00	1.00	1.00	1.00
REC	1.00	1.00	1.00	1.00	1.00
DEC	1.00	1.00	1.00	1.00	1.00
Total	1.27	1.47	1.26	1.00	2.37
Model 2	**BFC**	**RZ**	**DZ**	**Octree**	**All**
RP	1.44	486.87	1.00	1.00	683.60
DE	1.26	1.00	1.55	1.00	1.78
RC	1.00	1.00	1.00	1.00	1.00
DC	1.00	1.00	1.00	1.00	1.00
REC	1.00	1.00	1.00	1.00	1.00
DEC	1.00	1.00	1.00	1.00	1.00
Total	1.25	1.41	1.23	1.00	2.10
Model 3	**BFC**	**RZ**	**DZ**	**Octree**	**All**
RP	1.36	629.33	1.00	1.00	877.28
DE	1.21	1.00	1.53	1.00	1.72
RC	1.00	1.00	1.00	1.00	1.00
DC	1.00	1.00	1.00	1.00	1.00
REC	1.00	1.00	1.00	1.00	1.00
DEC	1.00	1.00	1.00	1.00	1.00
Total	1.20	1.38	1.23	1.00	1.99

Table D.8: Results for shadow tests using simple rays for the original program, back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), octree and for all optimisations. The values are the average, maximum and minimum number of intersections per ray. The reduction factor compared to the original program is given for the average number of intersections.

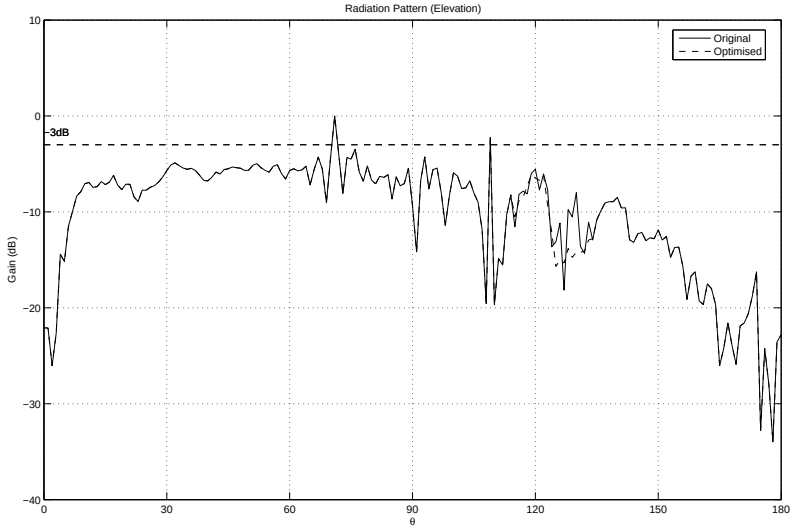
Model 1	**Original**	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Average	67.19	72.93	67.19	73.14	6.77	6.67
Reduction		**0.92**	**1.00**	**0.92**	**9.93**	**10.07**
Maximum	82	82	82	82	33	33
Minimum	1	1	1	1	0	0
Model 2	**Original**	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Average	106.45	111.60	106.45	111.99	5.45	5.39
Reduction		**0.95**	**1.00**	**0.95**	**19.53**	**19.76**
Maximum	125	125	125	125	40	40
Minimum	1	1	1	1	0	0
Model 3	**Original**	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Average	142.50	147.78	142.50	148.40	6.09	6.07
Reduction		**0.96**	**1.00**	**0.96**	**23.39**	**23.47**
Maximum	169	169	169	169	42	42
Minimum	1	1	1	1	0	0

Table D.9: Mean absolute error (m_{ae}) for simple rays. The overall error and error in each radiation pattern is shown for back face culling (BFC), diffraction zones (DZ) and when all optimisations are used.

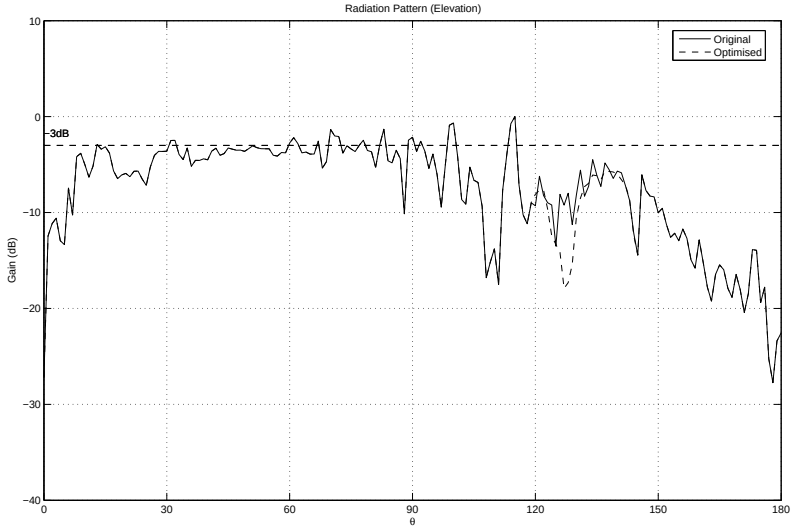
Model 1	**BFC**	**DZ**	**Overall**
m_{ae}	0 dB	1.95×10^{-2} dB	1.95×10^{-2} dB
Model 2	**BFC**	**DZ**	**Overall**
m_{ae}	0 dB	1.92×10^{-2} dB	1.92×10^{-2} dB
Model 3	**BFC**	**DZ**	**Overall**
m_{ae}	0 dB	1.97×10^{-2} dB	1.97×10^{-2} dB



(a) Radiation pattern of $\theta = 0°..180°$, $\phi = 131°$ for model 1. $\mathrm{m_{ae}} = 2.36{\times}10^{-1}$ dB



(b) Radiation pattern of $\theta = 0°..180°$, $\phi = 137°$ for model 2. $\mathrm{m_{ae}} = 1.91{\times}10^{-1}$ dB

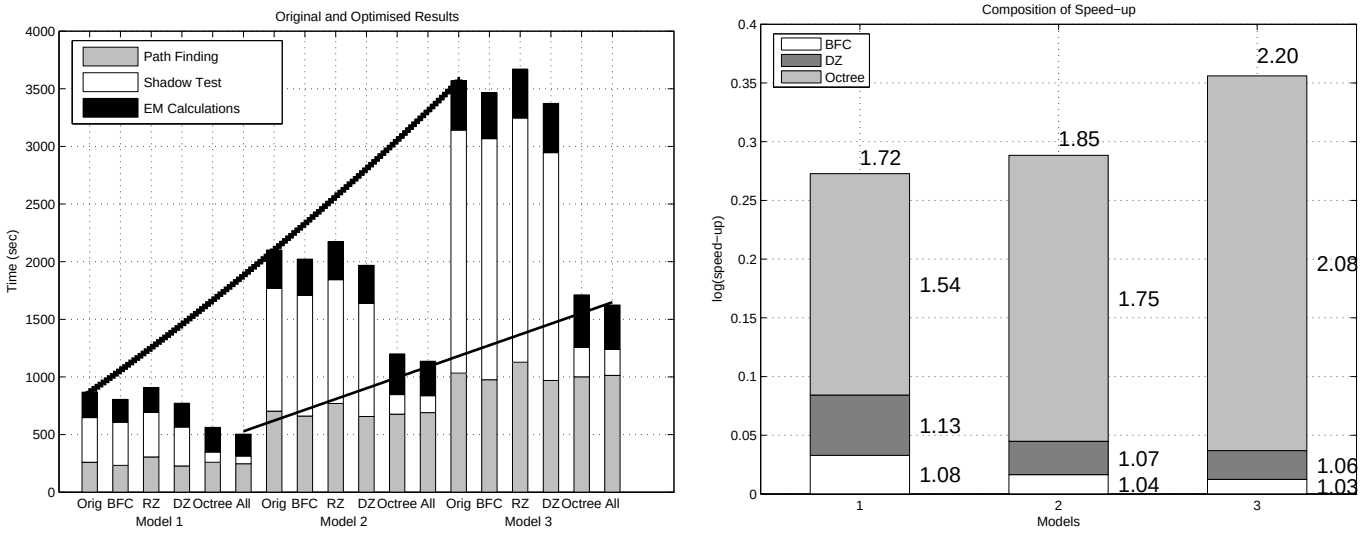


(c) Radiation pattern of $\theta = 0°..180°$, $\phi = 131°$ for model 3. $\mathrm{m_{ae}} = 2.58{\times}10^{-1}$ dB

Figure D.1: Radiation patterns for all 3 models using simple rays.

D.3.3 Discussion for Simple Rays

The results in the previous section are combined to present an overall picture of the speed-up and error due to the optimisation techniques. First the performance increase is examined. Then the performance increase and the introduced error are examined.



(a) Run times.

(b) Decomposition of speed-up.

Figure D.2: Run time and speed-up graphs for simple rays.

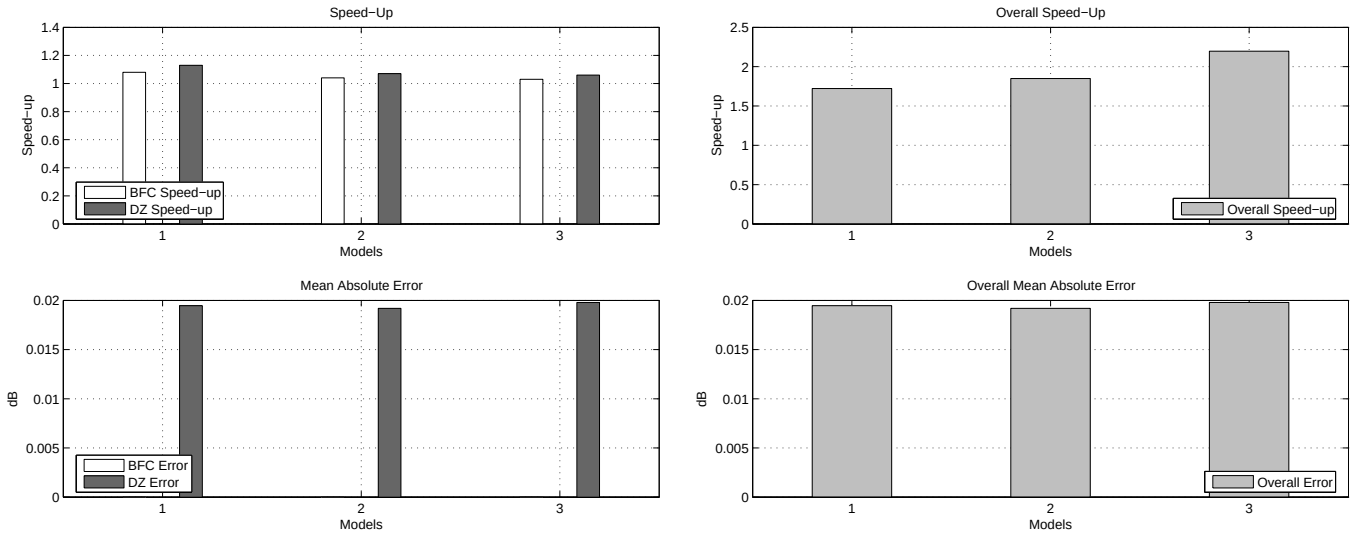
Figure D.2(a) shows the run time for each model. A group of 6 bars shows the results for one model. For each model the run time of the original program, of the 4 optimisations used individually and when all optimisations are combined are shown by the bars. Each bar is decomposed into the time for path finding, the shadow test time and the electromagnetic calculations. Trend lines show the increase in run time for the original program and the program using all optimisations.

The trend for the original program increases quadratically. For the optimised program the increase is linear. This is explained by the fact that the largest reduction in the run time is due to the octree. With the octree the shadow tests scale with model size. Examining the bars shows that the 3 path finding techniques do not reduce the run time by much. In contrast the octree significantly reduces the run time and leads to the overall reduction in the run time which now increases linearly from model 1 to model 3.

In figure D.2(b) the contribution of each optimisation technique to the overall speed-up is shown. Appendix D.7 explains how the graph is constructed. The values next to the bars are the speed-ups due to each optimisation. On top of the bar is the overall speed-up. The reflection zones slightly increase the run time because of the overhead in creating and accessing the RZ. In figure D.2(b), the results for RZ are therefore not shown.

The graph shows that the octree is responsible for the largest performance increase. The octree reduces the run time by a factor between 1.54 and 2.08. Diffraction zones reduce the run time by between 1.13 and 1.06. Back face culling reduces the run time by between 1.08 and 1.03. The run time using simple rays is dominated by the shadow tests. As the model size increases the octree provides a greater increase in performance because the run time of the original program increases exponentially with model size. Using the octree the shadow tests scale with model size. This leads to the large speed-up factors using the octree.

In figure D.3, the top 2 graphs shown the speed-up and the bottom 2 show the mean absolute error. On the left in figure D.3(a) the speed-up and error for back face culling and diffraction zones are shown. Only results for these 2 techniques are shown because they introduce errors into the radiation patterns. The speed-up remains relatively constant for all 3 models. For the 3 models back face culling introduces no errors. The error using the diffraction zones for each model is approximately 0.02 dB. No relationship between the performance and the magnitude of the error exists. The error and the speed-up are dependent on the geometry



(a) Speed-up and error comparison for BFC and DZ.

(b) Speed-up and error comparison for combination.

Figure D.3: Speed-up and error comparisons.

of the model. In figure D.3(b) the speed-up and error when all optimisations are enabled are shown. Again no relationship between the error and the speed-up exists.

D.4 Combined

D.4.1 Ray Tracing

Table D.10: Run times for models 1 to 3 using simple and second order rays for the original (orig) program, using back face culling (BFC), using reflection zones (RZ), using diffraction zones (DZ), using the octree and using all optimisations. All values in seconds.

Model 1	**Orig**	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Path finding	2768.37	2272.99	2612.86	1918.87	2765.21	1324.78
Shadow Time	217.31	204.15	213.31	178.03	47.09	33.43
EM	85.18	75.85	73.43	75.42	86.65	51.15
Overall Run Time	**3070.86**	**2553.00**	**2899.73**	**2172.42**	**2898.95**	**1409.58**
Model 2	**Orig**	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Total	8681.64	7423.53	8236.45	6173.90	8719.52	4679.20
Shadow Time	419.63	390.82	412.45	339.05	74.29	48.99
EM	151.28	127.22	122.67	130.02	151.62	87.08
Overall Run time	**9252.55**	**7941.59**	**8771.81**	**6643.16**	**8945.44**	**4815.58**
Model 3	**Orig**	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Total	18625.46	16406.84	17852.46	13408.81	18716.56	10573.13
Shadow Time	767.14	723.86	746.84	620.10	108.14	72.20
EM	256.72	219.98	204.21	219.47	259.59	143.55
Overall Run Time	**19649.33**	**17350.71**	**18803.91**	**14248.60**	**19084.31**	**10789.41**

Table D.10 lists the run times for each model. Vertically the table lists the time for path finding, shadow test, electromagnetic calculations and the overall time. Horizontally results are for the original program, the 4 individual techniques and the combination of the techniques.

The individual and collective speed-ups are given in tables D.11 to D.13 for path finding, the shadow tests and the overall run time. The path finding speed-up is shown individually for each ray. The total path finding speed-up is given at the end of the path finding section. The tables show that when combining all optimisations, path finding time is reduced by a factor of 1.9 on average, the shadow tests are 9 times faster on average and the overall run time is reduced by a factor of approximately 2, ie. the optimised program runs in half the time of the original program. In model 3 for example, the path finding time is reduced by

Table D.11: Speed-up factors for model 1 for back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), the octree and all optimisations.

	BFC	RZ	DZ	Octree	All
Path Finding					
RP	2.49	–	1.99	1.12	–
DE	1.41	1.11	1.98	1.09	1.60
RC	0.76	0.76	0.65	1.31	0.87
DC	0.72	1.25	1.72	1.01	1.30
REC	1.00	1.00	1.00	0.34	0.34
DEC	1.06	1.38	1.16	0.85	1.02
RP-RP	1.41	1270.45	1.00	1.01	–
RP-DE	1.42	10.52	1.51	0.99	25.99
DE-RP	1.32	1.01	1.30	1.00	1.65
DE-DE	1.28	1.00	1.80	1.00	2.69
RC-RC	1.02	0.98	1.00	0.96	0.99
RC-DC	1.30	0.83	0.98	1.05	0.84
DC-RC	1.38	0.69	3.48	0.70	2.35
DC-DC	1.25	1.16	0.98	1.67	1.00
DL-DL	1.00	0.95	0.86	1.02	0.97
RP-RC	1.45	22.28	1.02	0.99	34.43
RP-DC	1.40	10.17	1.03	1.07	24.34
DE-RC	1.22	1.00	1.09	1.00	1.34
DE-DC	1.26	1.01	1.08	0.99	1.39
RC-RP	0.95	0.98	1.00	1.01	1.01
RC-DE	1.00	1.00	1.42	1.00	1.71
DC-RP	1.00	0.95	0.99	1.10	1.09
DC-DE	0.99	1.00	1.42	0.98	1.64
Total	1.22	1.06	1.44	1.00	2.09
Shadow Time	1.06	1.02	1.22	4.61	7.29
Overall Run Time	1.20	1.06	1.41	1.06	2.18

a factor of 1.76 from 5 hours and 10 minutes in the original program to just under 3 hours using all optimisations.

The path finding times for DE-DE, DE-RC and RC-DE which make up the largest part of the path finding time are reduced by 1.9 on average. This means these rays run in just under half the original time. These 3 ray types take up over 80 % of the path finding time and therefore the speed-up for these rays are the most important [2].

Although the shadow time is reduced by a factor of 9 using the octree, the shadow time is negligible when compared with the total run time (see the run times in table D.10). Again BFC and DZ also contribute slightly to the reduction in the shadow time.

The reductions in the search space are given in tables D.14 to D.16. The overall reduction in the search space is given in the last row in the tables. Using all the optimisations the number of rays that are traced is reduced by a factor of 2.77 on average. Or put another way, the number of rays using all optimisations is reduced by 64 %. Although this is a large reduction many of the ray-types have a small path finding time.

The shadow results for the tests are listed in table D.17. The shadow optimisations reduce the average number of intersection tests per ray by a factor of 9 for model 1, by 13 for model 2 and by 16 for model 3. The maximum number of tests are reduced to less than half the number of components in each model. The performance increase is due to the fact that the octree is *aware* of the geometric structure of the model. The larger the model the longer the run time of the brute force algorithm in the original program and the more effective the octree algorithm becomes. These results are very similar to the results for simple rays in section D.3. This indicates that the octree works independent of model size and of the ray types.

D.4.2 Errors

Table D.18 lists the errors for models 1 to 3 using simple and second order rays. For each model, the table is divided into 4 parts. Three radiation patterns are calculated for each model. The overall error in all 3 radiation patterns is listed first. The next 3 sections list the errors in each radiation pattern. In each case the mean absolute error (m_{ae}) is given.

Table D.12: Speed-up factors for model 2 for back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), the octree and for all optimisations.

	BFC	RZ	DZ	Octree	All
Path Finding					
RP	1.33	–	0.70	1.11	–
DE	1.31	0.94	1.99	1.16	2.83
RC	0.67	0.89	0.94	0.70	1.01
DC	1.77	1.55	0.95	1.22	1.21
REC	2.57	1.26	4.81	0.62	1.64
DEC	1.01	1.19	1.13	1.10	1.12
RP-RP	1.43	4098.42	1.00	1.01	4235.03
RP-DE	1.47	19.22	1.56	0.99	39.43
DE-RP	1.32	1.00	1.22	0.98	1.53
DE-DE	1.30	1.00	1.78	0.99	2.65
RC-RC	1.01	1.00	1.00	0.99	0.99
RC-DC	0.96	0.97	1.01	1.00	1.00
DC-RC	0.96	1.15	0.95	1.03	1.03
DC-DC	0.97	1.02	0.99	0.95	0.93
DL-DL	0.88	0.90	0.91	0.92	0.99
RP-RC	1.46	33.06	1.00	1.00	58.66
RP-DC	1.43	29.17	1.05	1.04	44.63
DE-RC	1.23	1.00	1.25	1.00	1.46
DE-DC	1.27	1.01	1.29	1.01	1.54
RC-RP	1.00	1.01	1.00	1.01	1.01
RC-DE	1.00	1.00	1.45	1.00	1.65
DC-RP	0.98	0.98	0.97	0.95	0.98
DC-DE	1.00	0.99	1.41	0.99	1.60
Total	1.17	1.05	1.41	1.00	1.86
Shadow Time	1.07	1.02	1.24	5.65	9.02
Overall Run time	1.17	1.05	1.39	1.03	1.92

Errors are calculated for back face culling, diffraction zones and when all optimisations are used.

In general the errors for BFC are less than for DZ. More rays are optimised using DZ and therefore a higher error is to be expected. For the individual radiation patterns the mean absolute error is between 1.31×10^{-1} dB and 2.50×10^{-1} dB and the overall errors in all radiation patterns are between 1.42×10^{-1} dB and 1.90×10^{-1} dB.

In figure D.4 the radiation pattern for the original and optimised programs with the highest mean absolute error is plotted for all 3 models. The error is localised to a small region of the plot (between $\theta = 120°$ and $\theta = 160°$).

Table D.13: Speed-up factors for model 3 for back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), the octree and for all optimisations.

	BFC	RZ	DZ	Octree	All
Path Finding					
RP	1.41	–	1.19	0.94	–
DE	1.66	1.23	1.81	1.15	2.30
RC	1.37	1.01	1.10	0.96	1.19
DC	0.96	1.47	0.98	1.12	0.88
REC	2.03	0.98	1.00	1.97	1.27
DEC	0.98	0.95	0.95	1.05	1.04
RP-RP	1.36	1352.06	1.00	1.00	5167.87
RP-DE	1.36	24.59	1.49	0.99	54.74
DE-RP	1.22	0.98	1.21	0.98	1.47
DE-DE	1.23	0.99	1.84	0.99	2.53
RC-RC	0.99	0.99	0.99	0.99	0.99
RC-DC	0.99	1.00	1.00	0.99	1.00
DC-RC	1.03	0.95	0.96	1.00	0.97
DC-DC	1.06	1.04	0.99	1.10	0.98
DL-DL	0.98	0.97	0.95	0.93	0.91
RP-RC	1.36	40.99	1.00	0.99	57.31
RP-DC	1.35	35.56	0.99	1.00	44.87
DE-RC	1.19	1.00	1.21	1.00	1.39
DE-DC	1.21	1.00	1.23	0.99	1.43
RC-RP	1.01	1.01	1.01	1.01	1.01
RC-DE	1.00	1.00	1.44	1.00	1.61
DC-RP	1.01	0.99	1.00	1.00	1.00
DC-DE	1.00	1.00	1.43	1.00	1.57
Total	1.14	1.04	1.39	1.00	1.76
Shadow Time	1.06	1.03	1.24	7.09	11.11
Overall Run Time	1.13	1.04	1.38	1.03	1.82

Table D.14: Search space reduction for model 1 using simple and second order rays for back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), octree and for all optimisations.

	BFC	RZ	DZ	Octree	All
RP	1.42	275.99	1.00	1.00	426.59
DE	1.25	1.00	1.51	1.00	1.74
RC	1.00	1.00	1.00	1.00	1.00
DC	1.00	1.00	1.00	1.00	1.00
REC	1.00	1.00	1.00	1.00	1.00
DEC	1.00	1.00	1.00	1.00	1.00
RP-RP	1.42	3119.89	1.00	1.00	5706.38
RP-DE	1.43	14.77	1.41	1.00	31.09
DE-RP	1.25	1.00	1.24	1.00	1.51
DE-DE	1.25	1.00	1.73	1.00	2.43
RC-RC	1.00	1.00	1.00	1.00	1.00
RC-DC	1.00	1.00	1.00	1.00	1.00
DC-RC	1.00	1.00	1.00	1.00	1.00
DC-DC	1.00	1.00	1.00	1.00	1.00
DL-DL	1.00	1.00	1.00	1.00	1.00
RP-RC	1.42	16.72	1.00	1.00	22.86
RP-DC	1.42	16.72	1.00	1.00	22.86
DE-RC	1.25	1.00	1.08	1.00	1.35
DE-DC	1.25	1.00	1.08	1.00	1.35
RC-RP	1.00	1.00	1.00	1.00	1.00
RC-DE	1.00	1.00	1.43	1.00	1.67
DC-RP	1.00	1.00	1.00	1.00	1.00
DC-DE	1.00	1.00	1.43	1.00	1.67
Total	1.30	1.48	1.39	1.00	2.90

Table D.15: Search space reduction for model 2 using simple and second order rays for back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), octree and for all optimisations.

	BFC	Reflection	Diffraction	Octree	All
RP	1.44	387.98	1.00	1.00	606.26
DE	1.26	1.00	1.54	1.00	1.77
RC	1.00	1.00	1.00	1.00	1.00
DC	1.00	1.00	1.00	1.00	1.00
REC	1.00	1.00	1.00	1.00	1.00
DEC	1.00	1.00	1.00	1.00	1.00
RP-RP	1.44	5409.69	1.00	1.00	9391.12
RP-DE	1.44	23.45	1.45	1.00	53.86
DE-RP	1.26	1.00	1.19	1.00	1.46
DE-DE	1.26	1.00	1.72	1.00	2.43
RC-RC	1.00	1.00	1.00	1.00	1.00
RC-DC	1.00	1.00	1.00	1.00	1.00
DC-RC	1.00	1.00	1.00	1.00	1.00
DC-DC	1.00	1.00	1.00	1.00	1.00
DL-DL	1.00	1.00	1.00	1.00	1.00
RP-RC	1.44	31.96	1.00	1.00	46.47
RP-DC	1.44	31.96	1.00	1.00	46.47
DE-RC	1.26	1.00	1.27	1.00	1.52
DE-DC	1.26	1.00	1.27	1.00	1.52
RC-RP	1.00	1.00	1.00	1.00	1.00
RC-DE	1.00	1.00	1.45	1.00	1.65
DC-RP	1.00	1.00	1.00	1.00	1.00
DC-DE	1.00	1.00	1.45	1.00	1.65
Total	1.30	1.49	1.36	1.00	2.77

Table D.16: Search space reduction for model 3 using simple and second order rays for back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), octree and for all optimisations.

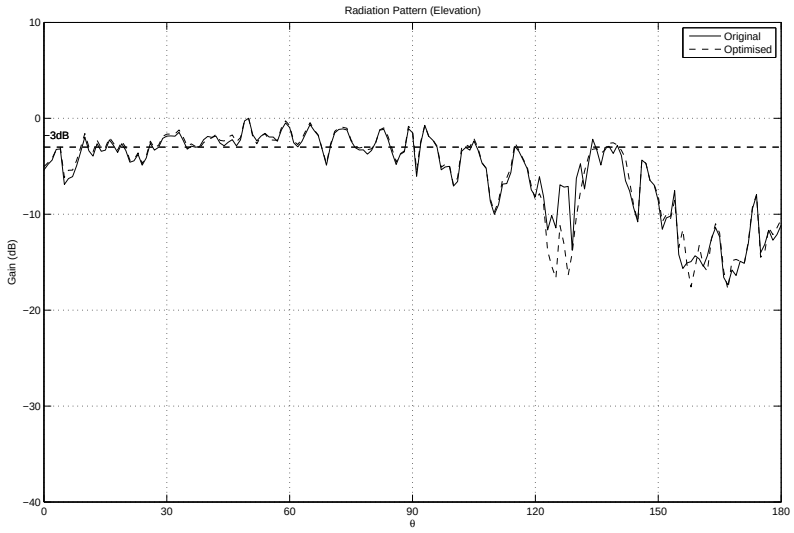
	BFC	Reflection	Diffraction	Octree	All
RP	1.36	490.23	1.00	1.00	741.66
DE	1.21	1.00	1.52	1.00	1.70
RC	1.00	1.00	1.00	1.00	1.00
DC	1.00	1.00	1.00	1.00	1.00
REC	1.00	1.00	1.00	1.00	1.00
DEC	1.00	1.00	1.00	1.00	1.00
RP-RP	1.36	8631.05	1.00	1.00	13877.00
RP-DE	1.36	33.72	1.46	1.00	74.10
DE-RP	1.21	1.00	1.20	1.00	1.42
DE-DE	1.21	1.00	1.78	1.00	2.33
RC-RC	1.00	1.00	1.00	1.00	1.00
RC-DC	1.00	1.00	1.00	1.00	1.00
DC-RC	1.00	1.00	1.00	1.00	1.00
DC-DC	1.00	1.00	1.00	1.00	1.00
DL-DL	1.00	1.00	1.00	1.00	1.00
RP-RC	1.36	38.45	1.00	1.00	51.12
RP-DC	1.36	38.45	1.00	1.00	51.12
DE-RC	1.21	1.00	1.23	1.00	1.43
DE-DC	1.21	1.00	1.23	1.00	1.43
RC-RP	1.00	1.00	1.00	1.00	1.00
RC-DE	1.00	1.00	1.44	1.00	1.60
DC-RP	1.00	1.00	1.00	1.00	1.00
DC-DE	1.00	1.00	1.44	1.00	1.60
Total	1.24	1.46	1.39	1.00	2.64

Table D.17: Results for shadow tests using simple and second order rays for the original program, for back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), octree and for all optimisations. The values are the average, maximum and minimum number of intersections per ray. The reduction factor compared to the original program is given for the average number of intersections.

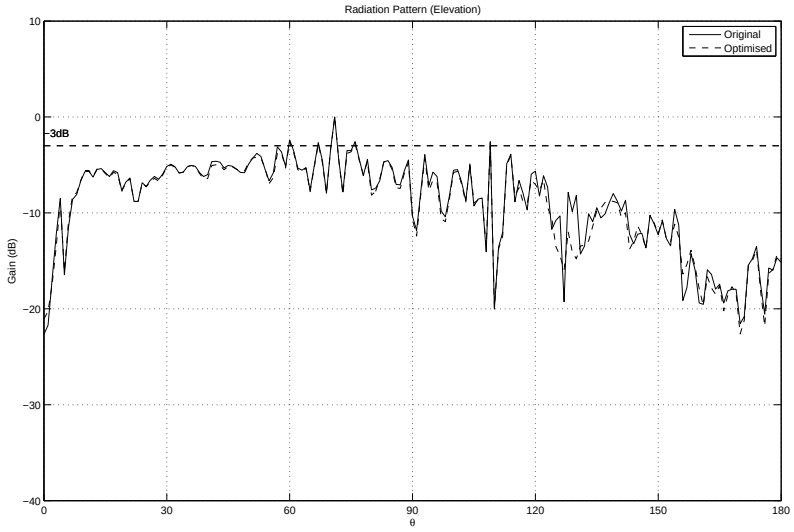
Model 1	**Original**	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Average	64.43	69.98	64.35	65.59	6.63	6.87
Reduction		**0.92**	**1.00**	**0.98**	**9.72**	**9.37**
Maximum	82	82	82	82	32	32
Minimum	1	1	1	1	0	0
Model 2	**Original**	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Average	94.92	102.77	94.49	97.08	6.74	7.14
Reduction		**0.92**	**1.00**	**0.98**	**14.08**	**13.29**
Maximum	125	125	125	125	38	38
Minimum	1	1	1	1	0	0
Model 3	**Original**	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Average	120.18	130.68	119.56	122.52	7.04	7.24
Reduction		**0.92**	**1.01**	**0.98**	**17.08**	**16.61**
Maximum	169	169	169	169	42	42
Minimum	1	1	1	1	0	0

Table D.18: Mean absolute errors (m_{ae}) using simple and second order rays. The overall error and error in each radiation pattern is shown for back face culling (BFC), diffraction zones (DZ) and when all optimisations are used.

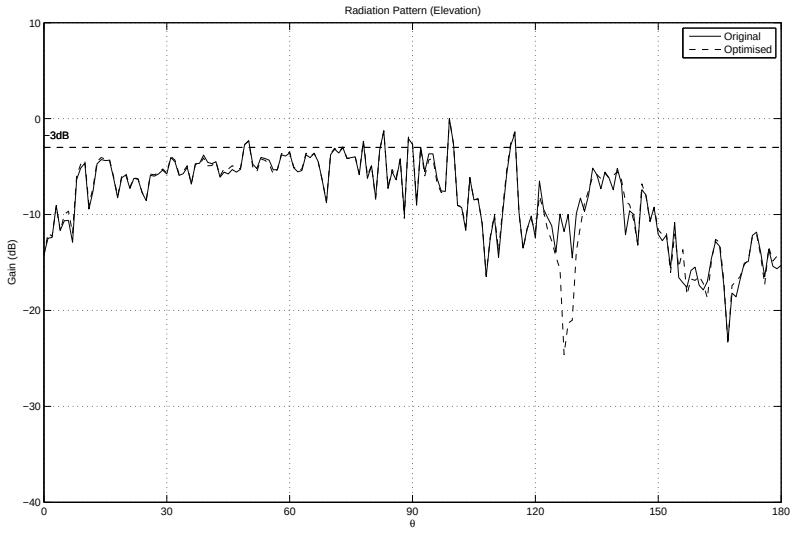
Model 1	**BFC**	**DZ**	**All**
	Overall error in all radiation patterns		
m_{ae}	3.65×10^{-2} dB	5.34×10^{-2} dB	1.42×10^{-1} dB
	Radiation pattern: $\theta = 90°,\ \phi = 0°..360°$		
m_{ae}	3.65×10^{-2} dB	2.36×10^{-2} dB	1.34×10^{-1} dB
	Radiation pattern: $\theta = 0°..360°,\ \phi = 0°$		
m_{ae}	3.03×10^{-2} dB	6.21×10^{-2} dB	1.62×10^{-1} dB
	Radiation pattern: $\theta = 0°..360°,\ \phi = 90°$		
m_{ae}	4.36×10^{-2} dB	7.30×10^{-2} dB	1.31×10^{-1} dB
Model 2	**BFC**	**DZ**	**All**
	Overall error in all radiation patterns		
m_{ae}	3.19×10^{-2} dB	7.54×10^{-2} dB	1.90×10^{-1} dB
	Radiation pattern: $\theta = 90°,\ \phi = 0°..360°$		
m_{ae}	3.41×10^{-2} dB	2.27×10^{-2} dB	1.63×10^{-1} dB
	Radiation pattern: $\theta = 0°..360°,\ \phi = 0°$		
m_{ae}	2.96×10^{-2} dB	1.07×10^{-1} dB	2.50×10^{-1} dB
	Radiation pattern: $\theta = 0°..360°,\ \phi = 90°$		
m_{ae}	2.84×10^{-2} dB	9.89×10^{-2} dB	1.56×10^{-1} dB
Model 3	**BFC**	**DZ**	**All**
	Overall error in all radiation patterns		
m_{ae}	3.14×10^{-2} dB	7.17×10^{-2} dB	1.80×10^{-1} dB
	Radiation pattern: $\theta = 90°,\ \phi = 0°..360°$		
m_{ae}	3.12×10^{-2} dB	3.28×10^{-2} dB	1.58×10^{-1} dB
	Radiation pattern: $\theta = 0°..360°,\ \phi = 0°$		
m_{ae}	2.75×10^{-2} dB	9.11×10^{-2} dB	2.32×10^{-1} dB
	Radiation pattern: $\theta = 0°..360°,\ \phi = 90°$		
m_{ae}	3.13×10^{-2} dB	9.38×10^{-2} dB	1.49×10^{-1} dB



(a) Radiation pattern of $\theta = 0°..180°$, $\phi = 131°$ for model 1. $\mathrm{m_{ae}} = 3.94{\times}10^{-1}$ dB

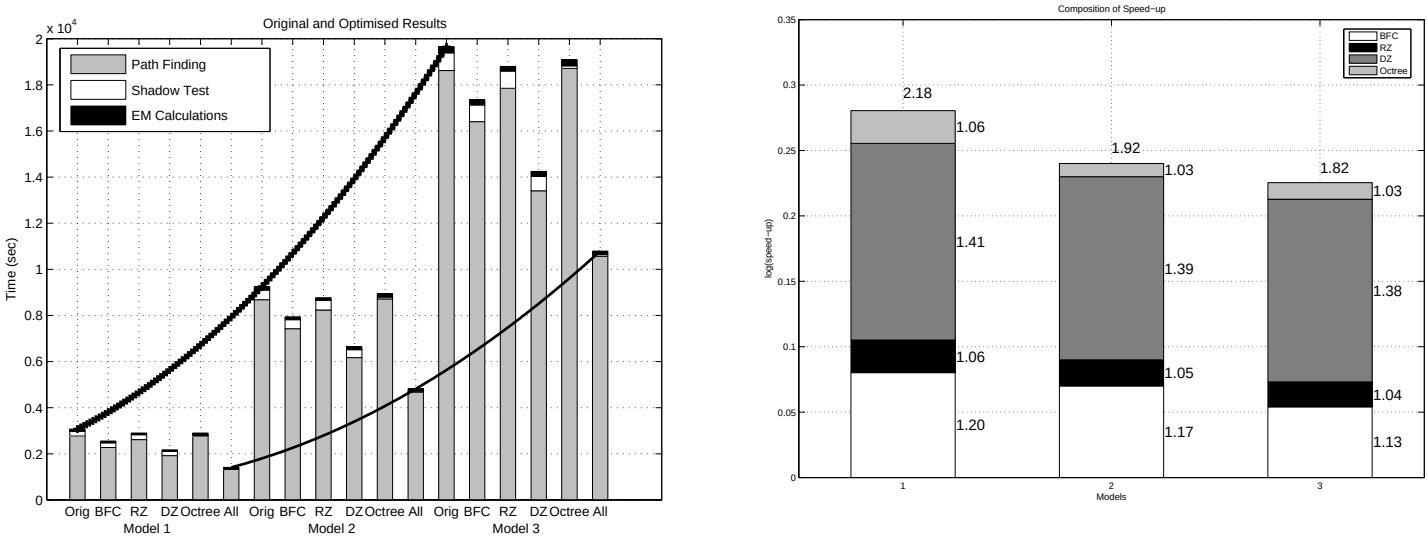


(b) Radiation pattern of $\theta = 0°..180°$, $\phi = 137°$ for model 2. $\mathrm{m_{ae}} = 3.74{\times}10^{-1}$ dB



(c) Radiation pattern of $\theta = 0°..180°$, $\phi = 131°$ for model 3. $\mathrm{m_{ae}} = 4.26{\times}10^{-1}$ dB

Figure D.4: Radiation patterns for all 3 models using simple and second order rays.



(a) Run time.

(b) Decomposition of speed-up.

Figure D.5: Run time and speed-up graphs.

D.4.3 Discussion for Combined Rays

The results from the 3 models presented in the previous section are combined to give an overall view of the performance and errors due to the optimisation techniques.

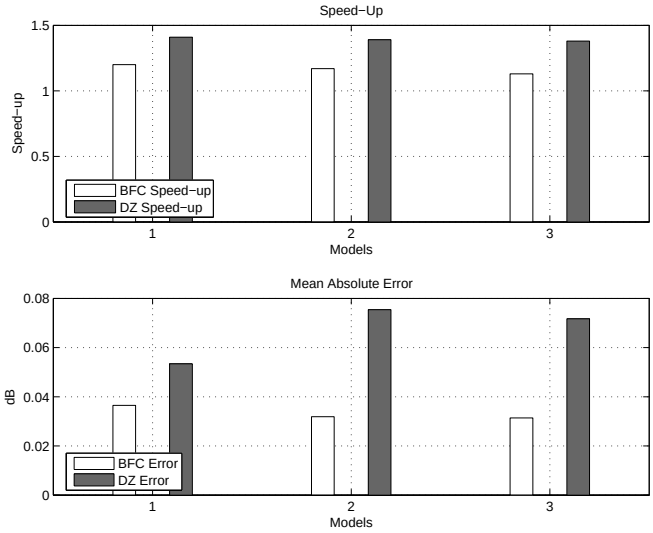
Figure D.5(a) shows the run time for each of the 3 models. A group of six bars represents the results for 1 model. The six bars show the run time for: the original program using no optimisations, only using back face culling, only using reflection zones, only using diffraction zones, only using the octree and using all 4 optimisations together. Each bar is divided into the time taken for path finding, for the shadow tests and for the electromagnetic calculations. Two trend curves are shown for the original program and the program using all the optimisations.

The figure shows that the run time of the original program increases exponentially. The trend for the optimised program is less steep but remains quadratic. Using all the optimisations the run time is reduced by approximately half. Path finding time takes up the largest part of the run time. Examining the bars, the path finding optimisations are responsible for the performance increase. The shadow time takes up a very small part of the run time and the octree cannot be used to reduce the run time when second order rays are used.

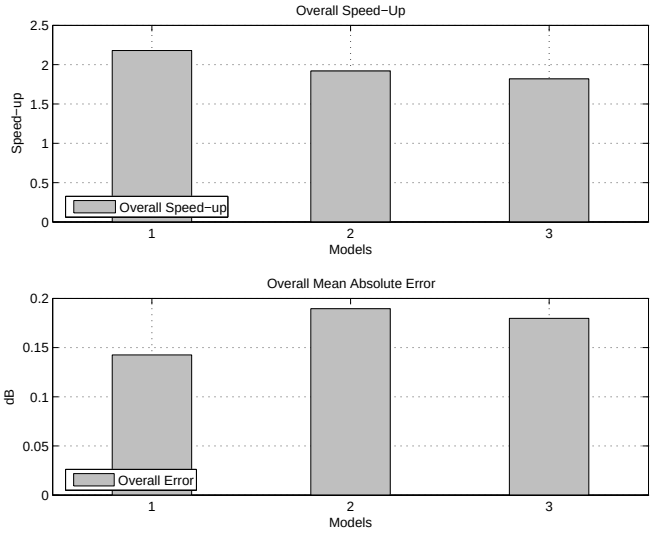
In figure D.5(b) the time speed-up for each model is shown. Each bar in the graph represents the overall speed-up for one model and shows the individual contributions of each optimisation technique. Appendix D.7 explains how the graphs are drawn. For example, for the first model the individual speed-ups obtained using back face culling, reflection zones, diffraction zones and the octree are 1.20, 1.06, 1.41 and 1.06 respectively. These values are shown to the right of each bar in figure D.5(b). The overall speed-up for model 1 is 2.18 which is shown on top of the bar.

The combination of the optimisations leads to speed-ups between 1.82 and 2.18. When using simple and second order rays back face culling and diffraction zones account for the largest performance increase. BFC reduces the run time by between 12 % and 17% and DZ reduce the original run time by approximately another 30 %. RZ and the octree both reduce the run time by a further 3 % to 5 %.

The large speed-up due to the diffraction zones is explained as follows. Many rays contain an edge diffraction and a ray impinging on an edge causes many diffracted rays. The diffraction zone can be used in many instances especially when optimising DE-DE, DE-RC and RC-DE rays which take up the largest part of path finding time. BFC also reduces the run time in a significant manner but is limited to polyhedrons. The reflection zones are limited to plate reflections and are not very effective at reducing the run time. The large amount of



(a) Speed-up and error comparison for BFC and DZ.



(b) Speed-up and error comparison for combination.

Figure D.6: Speed-up and error comparisons.

time spent on path finding means the shadow tests take up a comparatively small amount of time and the octree does not reduce the run time greatly.

The bars in figure D.5(b) show a downward trend with increasing model size. This is due to the fact that the search space is still very large even when the path finding optimisations are used. As shown by the trend line in figure D.5(a) for the optimisations, although the run time is reduced by the path finding optimisations, the increase in run time is non-linear which means the optimisations are less effective with increasing model size.

Figure D.6(a) shows the speed-up for back face culling and diffraction zones for the 3 models (top graph). Below that are shown the errors for the 2 techniques for each model. On the right in figure D.6(b) the overall speed-up and the error for each model are shown. Speed-up and error in the models do not shown any correlation. The error in the path finding optimisations is related to the geometry of the model and the model size and not to the speed-up.

The optimisations (back face culling and diffraction zones) introduce errors into the output. These errors are acceptable given the speed-up. The errors are localised to θ and ϕ ranges and in general the optimised output closely follows the original radiation patterns.

D.5 All Rays

In this section a model is tested using all rays avaliable in SuperNEC. A separate model is used to test all rays because the run time of tertiary rays is exorbitant. The model in this section is small enough to demonstrate that the optimisations work with tertiary rays.

D.5.1 Ray Tracing

The run time and speed-up factors are given in tables D.19 and D.20 respectively. Path finding time dominates and the time for the shadow tests and electromagnetic calculations is negligible. The shadow time is too small for the octree optimisations to be effective. The overall run time using all optimisations is reduced by a factor of 1.79. This is 44 % less than the original program.

Examining table D.19 shows that the tertiary rays take up most of the time. The path finding time for the tertiary rays takes up 92 % of the run time. Triple diffraction (DE-DE-DE) rays have the longest path finding time. Using BFC and DZ the run time of these rays is reduced by a factor of 1.90 which means that the path finding time of the original program is

Table D.19: Run times for all rays for the original (orig) program, back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), the octree and for all optimisations. All values in seconds.

	Orig	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Path Finding						
RP	0.05	0.02	0.02	0.03	0.02	0.00
DE	0.22	0.22	0.16	0.12	0.36	0.13
RC	0.18	0.42	0.22	0.20	0.38	0.31
DC	0.10	0.05	0.06	0.08	0.08	0.06
REC	0.02	0.00	0.03	0.02	0.02	0.00
DEC	0.89	0.71	0.83	0.92	0.70	0.77
RP-RP	3.55	2.00	0.03	3.63	4.09	0.00
RP-DE	5.49	2.83	1.52	4.31	5.39	1.40
DE-RP	4.37	3.37	4.32	4.06	4.52	3.24
DE-DE	84.50	65.38	84.38	64.86	84.80	49.22
RC-RC	7.30	7.37	7.10	7.33	6.73	6.63
RC-DC	0.92	0.89	0.66	1.07	0.71	0.85
DC-RC	0.14	0.09	0.10	0.06	0.03	0.11
DC-DC	0.14	0.30	0.11	0.20	0.25	0.12
DL-DL	2.11	2.19	1.95	1.64	2.00	2.08
RP-RC	5.64	3.23	1.51	5.98	5.68	0.69
RP-DC	1.57	0.61	0.25	1.23	1.61	0.23
DE-RC	117.45	93.77	117.18	106.33	117.30	92.73
DE-DC	7.09	5.93	6.96	6.75	7.53	5.53
RC-RP	6.05	6.11	5.94	5.88	5.82	5.88
RC-DE	152.15	151.66	153.00	137.19	152.90	124.87
DC-RP	1.31	1.47	1.39	1.50	1.53	1.64
DC-DE	5.95	5.67	5.83	5.29	5.49	4.94
RP-RP-RP	77.83	39.16	0.02	77.84	77.08	0.02
DE-RP-RP	78.78	60.19	78.95	69.26	78.72	57.85
DE-DE-RP	1613.45	1250.20	1614.18	1332.97	1615.34	997.58
DE-RP-DE	1443.05	1121.99	1442.63	1077.32	1445.07	832.93
DE-DE-DE	7291.30	5715.23	7289.41	5294.69	7293.28	3833.51
Total	**10980.62**	**8541.12**	**10837.55**	**8322.78**	**11026.67**	**6096.33**
Shadow Time	108.79	104.67	107.48	105.99	65.23	49.15
EM	3902.62	246.74	212.26	196.07	284.08	170.47
Overall Run Time	**11302.70**	**8892.45**	**11211.50**	**8627.20**	**11266.70**	**6315.91**

almost halved. The run time for triple reflection rays (RP-RP-RP), is significantly reduced using the RZ. However the run time of the RP-RP-RP rays in the original program is a little over a minute which is negligible when compared with DE-DE-DE rays which have a run time of approximately 2 hours. The other tertiary rays (DE-RP-RP, DE-DE-RP and DE-RP-DE) are optimised mainly using BFC and DZ. The path finding times for these rays are reduced by factors between 1.36 and 1.73.

The number of rays traced is shown in table D.21 and the reduction in the number of rays is given in table D.22. The results for simple and second order rays are similar to the results in sections D.3 and D.4. Examining table D.21 shows that in the original program for tertiary rays about 10 times more rays are traced than for simple and second order rays. The reduction in the run time is achieved mostly by reducing the run time of the tertiary rays. The overall search space reductions for tertiary rays correspond with the path finding time speed-ups given in the previous set of tables.

Results from the shadow tests are given in table D.23. The model in this section consists of 22 components and the brute force shadow tests run for less than 2 minutes. The optimised shadow tests run for 50 seconds. The average number of intersections per ray are reduced from 19 for the original program by a factor of 4.54 to about 4 intersections, using the octree.

Table D.20: Speed-up factors for all rays for back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), the octree and using all optimisations.

	BFC	RZ	DZ	Octree	All
Path Finding					
RP	3.13	2.94	1.47	2.94	–
DE	1.00	1.40	1.76	0.60	1.73
RC	0.44	0.85	0.91	0.49	0.59
DC	2.07	1.56	1.20	1.20	1.48
REC	–	0.53	1.00	1.07	–
DEC	1.25	1.07	0.96	1.26	1.15
RP-RP	1.78	111.03	0.98	0.87	–
RP-DE	1.94	3.60	1.27	1.02	3.92
DE-RP	1.30	1.01	1.08	0.97	1.35
DE-DE	1.29	1.00	1.30	1.00	1.72
RC-RC	0.99	1.03	1.00	1.08	1.10
RC-DC	1.03	1.40	0.87	1.31	1.09
DC-RC	1.53	1.49	2.33	4.73	1.31
DC-DC	0.48	1.31	0.72	0.56	1.15
DL-DL	0.96	1.09	1.29	1.05	1.02
RP-RC	1.75	3.74	0.94	0.99	8.20
RP-DC	2.56	6.32	1.28	0.98	6.76
DE-RC	1.25	1.00	1.10	1.00	1.27
DE-DC	1.20	1.02	1.05	0.94	1.28
RC-RP	0.99	1.02	1.03	1.04	1.03
RC-DE	1.00	0.99	1.11	1.00	1.22
DC-RP	0.89	0.94	0.87	0.86	0.80
DC-DE	1.05	1.02	1.13	1.08	1.20
RP-RP-RP	1.99	4864.13	1.00	1.01	4864.13
DE-RP-RP	1.31	1.00	1.14	1.00	1.36
DE-DE-RP	1.29	1.00	1.21	1.00	1.62
DE-RP-DE	1.29	1.00	1.34	1.00	1.73
DE-DE-DE	1.28	1.00	1.38	1.00	1.90
Total	1.28	1.00	1.31	1.00	1.79
Shadow Time	1.04	1.01	1.03	1.67	2.21
Overall Run Time	1.27	1.01	1.31	1.00	1.79

Table D.21: Number of rays traced for the model using all rays for the original program, back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), octree and for all optimisations.

	Orig	BFC	RZ	DZ	Octree	All
Direct	3850	3850	3850	3850	3850	3850
RP	77000	38500	1054	77000	77000	1029
DE	154000	123200	154000	117145	154000	111279
RC	7700	7700	7700	7700	7700	7700
DC	15400	15400	15400	15400	15400	15400
REC	15400	15400	15400	15400	15400	15400
DEC	30800	30800	30800	30800	30800	30800
RP-RP	1463000	731500	4223	1463000	1463000	1384
RP-DE	2926000	1447600	666050	2517060	2926000	399749
DE-RP	2926000	2340800	2926000	2648800	2926000	2279200
DE-DE	6006000	4804800	6006000	4714486	6006000	3882291
RC-RC	7700	7700	7700	7700	7700	7700
RC-DC	15372	15372	15372	15372	15372	15372
DC-RC	15372	15372	15372	15372	15372	15372
DC-DC	30744	30744	30744	30744	30744	30744
DL-DL	40043	40043	40043	40043	40043	40043
RP-RC	154000	77000	30800	154000	154000	15400
RP-DC	308000	154000	61600	308000	308000	30800
DE-RC	308000	246400	308000	277200	308000	242550
DE-DC	308000	246400	308000	277200	308000	242550
RC-RP	154000	154000	154000	154000	154000	154000
RC-DE	308000	308000	308000	272769	308000	256956
DC-RP	308000	308000	308000	308000	308000	308000
DC-DE	308000	308000	308000	272769	308000	256956
RP-RP-RP	30800000	15400000	10596	30800000	30800000	3240
DE-RP-RP	58520000	46816000	58520000	52976000	58520000	45584000
DE-DE-RP	114114000	91291200	114114000	95214350	114114000	76314700
DE-RP-DE	108416000	86744350	108416000	84350840	108416000	68087712
DE-DE-DE	246246000	196996800	246246000	183662353	246246000	140832615
Total	**573986381**	**448718931**	**539032704**	**460747353**	**573986381**	**339186792**

Table D.22: Search space reduction for the model using all rays for back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), octree and for all optimisations.

	BFC	**RZ**	**DZ**	**Octree**	**All**
Direct	1.00	1.00	1.00	1.00	1.00
RP	2.00	73.06	1.00	1.00	74.83
DE	1.25	1.00	1.31	1.00	1.38
RC	1.00	1.00	1.00	1.00	1.00
DC	1.00	1.00	1.00	1.00	1.00
REC	1.00	1.00	1.00	1.00	1.00
DEC	1.00	1.00	1.00	1.00	1.00
RP-RP	2.00	346.44	1.00	1.00	1057.08
RP-DE	2.02	4.39	1.16	1.00	7.32
DE-RP	1.25	1.00	1.10	1.00	1.28
DE-DE	1.25	1.00	1.27	1.00	1.55
RC-RC	1.00	1.00	1.00	1.00	1.00
RC-DC	1.00	1.00	1.00	1.00	1.00
DC-RC	1.00	1.00	1.00	1.00	1.00
DC-DC	1.00	1.00	1.00	1.00	1.00
DL-DL	1.00	1.00	1.00	1.00	1.00
RP-RC	2.00	5.00	1.00	1.00	10.00
RP-DC	2.00	5.00	1.00	1.00	10.00
DE-RC	1.25	1.00	1.11	1.00	1.27
DE-DC	1.25	1.00	1.11	1.00	1.27
RC-RP	1.00	1.00	1.00	1.00	1.00
RC-DE	1.00	1.00	1.13	1.00	1.20
DC-RP	1.00	1.00	1.00	1.00	1.00
DC-DE	1.00	1.00	1.13	1.00	1.20
RP-RP-RP	2.00	2906.76	1.00	1.00	9506.17
DE-RP-RP	1.25	1.00	1.10	1.00	1.28
DE-DE-RP	1.25	1.00	1.20	1.00	1.50
DE-RP-DE	1.25	1.00	1.29	1.00	1.59
DE-DE-DE	1.25	1.00	1.34	1.00	1.75
Total	**1.28**	**1.06**	**1.25**	**1.00**	**1.69**

Table D.23: Results for shadow tests using all rays for the original program, for back face culling (BFC), reflection zones (RZ), diffraction zones (DZ), octree and for all optimisations. The values are the average, maximum and minimum number of intersections per ray. The reduction factor compared to the original program is given for the average number of intersections.

	Original	**BFC**	**RZ**	**DZ**	**Octree**	**All**
Average	18.99	19.64	18.99	19.19	4.18	4.05
Reduction		**0.97**	**1.00**	**0.99**	**4.54**	**4.69**
Maximum	22	22	22	22	15	15
Minimum	1	1	1	1	0	0

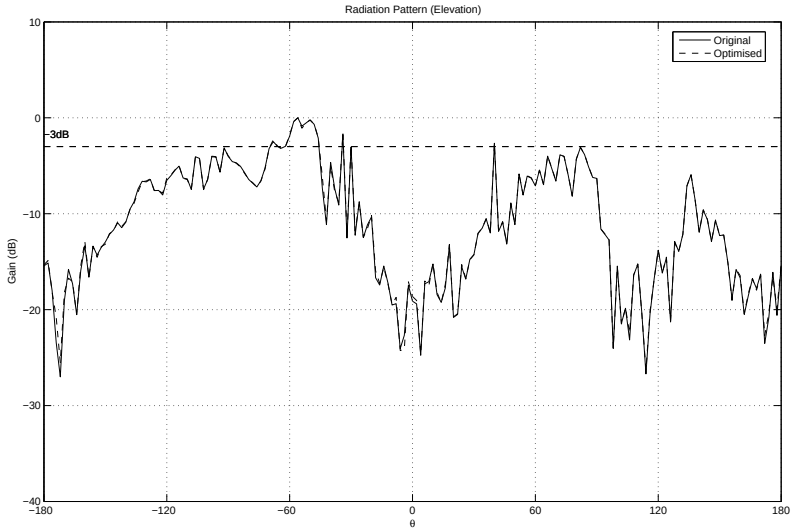


Figure D.7: Radiation pattern of $\theta = -180°..180°$ $\phi = 0°$ for tertiary model using all optimisations. $m_{ae} = 1.29 \times 10^{-1}$ dB

D.5.2 Errors

Table D.24: Mean absolute error (m_{ae}) using all rays. The overall error and error in each radiation pattern is shown for back face culling (BFC), diffraction zones (DZ) and when all optimisations are used.

	BFC	**DZ**	**All**
	Overall error in all radiation patterns		
m_{ae}	2.92×10^{-2} dB	1.57×10^{-2} dB	9.96×10^{-2} dB
	Radiation pattern: $\theta = 90°$, $\phi = 0°..360°$		
m_{ae}	1.63×10^{-2} dB	6.80×10^{-3} dB	3.51×10^{-2} dB
	Radiation pattern: $\theta = 0°..360°$, $\phi = 0°$		
m_{ae}	3.59×10^{-2} dB	1.72×10^{-2} dB	1.29×10^{-1} dB
	Radiation pattern: $\theta = 0°..360°$, $\phi = 90°$		
m_{ae}	3.66×10^{-2} dB	2.41×10^{-2} dB	1.28×10^{-1} dB

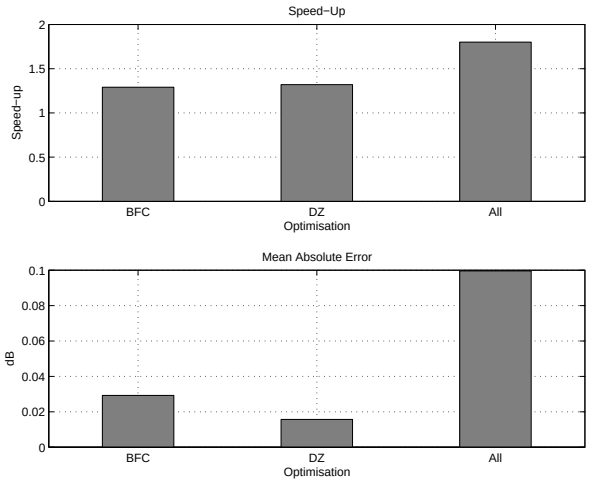
In table D.24 the errors introduced by back face culling and the diffraction zones are listed. The radiation pattern for the XZ-elevation plane ($\theta = -180°..180°$, $\phi = 0°$) which has the largest mean relative error is shown in figure D.7.

D.5.3 Discussion for All Rays

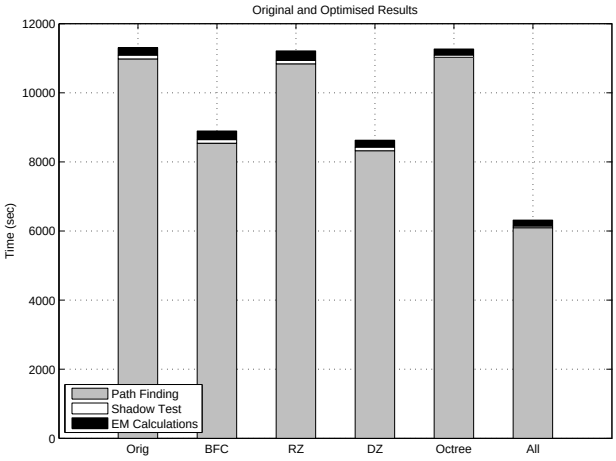
The optimisations also work with tertiary rays. However the run time of these rays is much higher. For the model the run time is reduced by a factor of 1.79. This is 44 % less time than the original program. The original program runs for approximately 3 hours. The optimised program runs for 1 hour and 45 minutes and introduces a mean absolute error of approximately 0.1 dB.

In figure D.8(a) the performance increase and the error are compared. The top graph shows the speed-up for BFC, DZ and all optimisations. The graph below that shows the error when using BFC, DZ and when using all optimisations.

In figure D.8(b) the run times of the program using the various optimisations are shown. Each bar is divided into path finding time, shadow test time and the time for electromagnetic calculations. The first bar is for the original program. Then follow the 4 optimisations individually. Only BFC and DZ reduce the run time in an appreciable manner. The final bar shows the run time when all optimisations are combined. Using BFC and DZ simultaneously, further reduces the run time. Both the shadow tests and the electromagnetic calculations take up an insignificant portion of the run time.



(a) Speed-up and error comparison for BFC, DZ and all optimisations.



(b) Run time with various optimisations.

Figure D.8: Speed-up and error comparisons.

D.6 Conclusion

The report has shown that for simple rays the optimisations reduce the run time for the example models by 50 % on average and the mean relative error is around 0.02 dB. For simple rays, the octree optimisations provide the largest reduction in the run time. The path finding optimisations do not significantly optimise the run time and can be disabled if only simple rays are used.

When second order and tertiary rays are used together with simple rays, the path finding optimisations are responsible for the largest reduction in the run time because of the large search space. The octree optimisations still reduce the shadow test time; however the shadow time is only a fraction of the path finding time. When using simple and higher order rays the run time for the example models is reduced by 45 % and the mean relative error is approximately 0.17 dB.

Out of the 3 path finding optimisations, the diffraction zones are the most effective. Many ray-types include an edge diffraction and edge diffraction leads to more than one ray which continues the propagation. The run time of the large number of edge diffracted rays is reduced by the diffraction zones.

The error is localised to small sections of the radiation patterns. There is no correlation between the speed-up and the error. The errors are caused by the geometric structure of the models.

D.7 Decomposition of Speed-up

In this section the bar graphs which show the contribution of the individual optimisations to the overall speed-up are explained.

The product of the individual speed-ups approximately equals the speed-up using all optimisations combined, ie.

$$S_{all} \approx S_{BFC} \cdot S_{RZ} \cdot S_{DZ} \cdot S_{octree} \tag{D.1}$$

Where S_{all}, S_{BFC}, S_{RZ}, S_{DZ} and S_{octree} are respectively the speed-ups using all optimisations, back face culling, reflection zones, diffraction zones and the octree. During the experiments, the overall speed-up was never exactly equal to the product of the individual optimisations. Every time the same simulation is run using the same optimisations and under the same conditions, there are slight variations in the run time because of the operating system and background processes.

Taking the logarithm on both sides of equation D.1

$$\log S_{all} \approx \log S_{BFC} + \log S_{RZ} + \log S_{DZ} + \log S_{octree} \tag{D.2}$$

Equation D.2 is plotted as a stacked bar graph to show the contribution of each optimisation to the overall performance increase. The y-axis for such a stacked bar graph is the logarithm of the speed-up.

References

[1] Intel pentium 4 processor. Intel Corporation. Last accessed 3 April 2006. [Online]. Available: http://www.intel.com/products/processor/pentium4/

[2] R. Hartleb, "Optimised ray tracing for the SuperNEC implementation of the uniform theory of diffraction," MSc(Eng) Dissertation, University of the Witwatersrand, Johannesburg, 2006, Appendix A: Analysis of the run time of SuperNEC-UTD: a MoM electromagnetic simulation package.

[3] ——, "Optimised ray tracing for the SuperNEC implementation of the uniform theory of diffraction," MSc(Eng) Dissertation, University of the Witwatersrand, Johannesburg, 2006, Appendix B: Path Finding Optimisations for SuperNEC-UTD.

Appendix E

Contents of CD

E.1 Contents of CD

The source code for the path finding optimisations and the shadow optimisations are in folders `PathFinding/` and `Octree/` respectively.

E.1.1 Folder: `PathFinding/`

- **Source Code**
  - **Back Face Culling**
    - `polyhedronidenitfy.h` Header file for back face culling.
    - `polyhedronidentify.cpp` C++ file for back face culling. Identifies the polyhedrons in a model.
  - **Reflection Zones**
    - `reflectionbufferwrapper.h` Header file for wrapper class for reflection zones.
    - `reflectionbufferwrapper.cpp` C++ file for wrapper class for reflection zones. The class constructs all reflections zones for a model.
    - `reflectionbuffer.h` Header file for a reflection zone.
    - `reflectionbuffer.cpp` C++ file for a reflection zone. This class constructs a single reflection zone for one plate and one segment.
  - **Diffraction Zones**
    - `diffractionbuffer.h` Header file for diffraction zones.
    - `diffractionbuffer.cpp` C++ file for diffraction zones. This class constructs a diffraction zone for an edge.
- **Unit Testing** The files used in the unit tests for path finding are listed below.
  - `main.cpp` Starts the unit tests for back face culling, reflection zones and diffraction zones.
  - `polyhedronidentifytest.h` The unit tests for back face culling.
  - `reflectionbuffertest.h` The unit tests for reflection zones.
  - `diffractionbuffertest.h` The unit tests for diffraction zones.
  - The `*.nec` files in the `PathFinding/` folder are required by the unit tests.

E.1.2 Folder: `Octree/`

- **Source Code**
  - `voxel2.h` File contains declarations for the `Voxel2` class for the octree.
  - `voxel2.cpp` File contains definitions of the `Voxel2` class for the octree.
  - `octree2.h` File contains declarations for the octree (`Octree2` class).
  - `octree2.cpp` File contains definitions of the octree (`Voxel2` class).
- **Unit Testing** The files used in the unit testing for the octree are listed below.
  - `main.cpp` Starts the unit tests for the octree.
  - `octree_unittest2.h` All the unit tests for the octree.
  - All `*.nec` files in folder `Octree/for_unit_tests/` are required by the octree unit tests.